

Visual Perception for Robotics Supporting Few-Shot Learning

Zur Erlangung des akademischen Grades eines
Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte
Dissertation

von

M. Sc.

Philip Keller

aus Durham

Tag der mündlichen Prüfung:
Erster Gutachter:
Zweiter Gutachter:

23.07.2026
Prof. Dr.-Ing. Rüdiger Dillmann
Prof. Dr.techn. Markus Vincze

Acknowledgments

I conducted this thesis while working at the FZI Research Center for Information Technology as a research scientist and pursuing my doctoral studies at the Karlsruhe Institute of Technology. In such an active, multidisciplinary research environment, I had the opportunity to meet many interesting and talented people. In this section, I would like to express my appreciation to them.

First and foremost, I would like to begin with my supervisor, Prof. Rüdiger Dillmann. Without him, this dissertation would have come to a stop on multiple occasions. Thanks to him I learned about the FZI and could work on the topic that fascinated me the most. Thanks to his experience in supervising PhD students, he guided me through difficult times and helped me finally perform a *landing* on the PhD runway.

My sincere thanks also go to my second examiner, Prof. Markus Vincze, for agreeing to review this dissertation. I greatly appreciate the time, care, and expertise dedicated to evaluating my work.

Large thanks also go to my first group leader, Arne Roennau, as well as to the subsequent group leaders and co-leaders, Georg Heppner, Tristan Schnell, and David Timmermann. They ensured that I could follow my scientific curiosity and continuously provided valuable feedback along the way. In particular, Arne's advice and concrete plans helped me navigate one or two challenging moments.

A big thank you also goes to all my current and former colleagues. Working on such a wide range of research projects was made even more enjoyable by their collaboration and support. And I am still impressed by all their individual skills, which I tried to learn from as much as possible. I joined the FZI as a student at the same time as Jakob Weinland, and we then continued working together for multiple years. Having started as a research scientist earlier than me, he helped a lot to get on board in the beginning. His humor is one of a kind and I am glad that he is still around in Karlsruhe after leaving FZI. My Master's thesis supervisor, Jacques Kaiser, initially encouraged me to pursue a PhD, and I later became his colleague. On top of that, he gave me a lot of inspiration: his amiable character, being super smart while at the same time easy to get along with, and always taking life with a smile, even when facing difficult challenges. Although not in my research group, I had frequent scientific discussions with Karam Daaboul. His admirable enthusiasm for research spilled over, whereas his passion for running somehow did not. Together with Nicolas Hügel, a true expert in designing mechanical parts for any task you can think of and the best long-term office roommate I could have imagined, they made my office a very pleasant place, where only the COVID-19 pandemic restrictions kept me from going there. Instead of running, for recovery and relaxation,

I preferred spending my free time sharing my passion for Volleyball and balcony barbecue with my colleagues Martina Overbeck, Vincenzo di Pentima, Sabine Bellmann, and Friedrich Graaf. Likewise, my former colleagues Atanas Tanev and Gabriele Bolano were not only great partners for discussions about robotics, but also wonderful companions on many trips, during which they taught me how to enjoy life through good food, wine, and nature. During my time at FZI, Johannes Mangler was almost constantly the project lead of at least one of my projects. I really liked his efficient way of supervising projects, and he also demonstrated multiple times how to manage larger projects with tight deadlines confidently. Two more people whose support, organizational overview, patience, and ability to restore order after bureaucratic chaos were quite essential to me having more capacity for other tasks: our secretaries Nicole Rösch and Sonja Göttl.

Joining me on my research path, my students and research assistants played a significant role. Naming them individually would span multiple lines, I still want to sincerely thank every one of them for their interest in my research and their support. Having the chance to supervise highly motivated and dedicated students was a very rewarding experience.

A huge thank you also to my parents, brothers, grandparents, family, and friends. They reminded me that there is something more important than work. They tolerated my unresponsiveness during busier times and consistently asked when I would finally finish, ensuring just the right amount of pressure. Last but not least, I want to thank my girlfriend Sandra for always lifting my spirits, her patience, and wholehearted support over the years.

Abstract

This dissertation addresses the challenge of enabling object-centric visual perception for robotic applications with constrained resources, such as limited training data and limited preparation time. Although fully supervised deep learning approaches achieve superior performance, their data- and compute-intensive preparation pipelines can limit their deployment in real-world robotic scenarios, particularly for tasks where time is critical or when operators are unfamiliar with machine learning methods. To overcome these limitations, this work investigates how pretrained vision models can serve as the foundation for perception systems that require only minimal task-specific training, and how the human operator can be involved to improve task alignment of the models.

The first part of this work focuses on resource-efficient perception models built on top of frozen pretrained backbones. Instead of relying on extensive training procedures, these approaches leverage prototypical representations and lightweight modular extensions to adapt to recognition or detection tasks from only a few examples. Different techniques for different levels of task granularity are explored, e.g., by incorporating part-based inference.

The second part extends the presented approaches to include interactive learning, where human operators can guide the model and adapt its behavior to the task at hand. By adjusting feature relevance or by directing the model's focus to important details, contextual knowledge can be transferred directly to the model without requiring additional training data. This can be particularly useful in underspecified few-shot scenarios, where data alone is insufficient to disambiguate the task.

Finally, the proposed methods are evaluated on robotics-oriented datasets and application scenarios. The results show that pretrained models offer diverse capabilities for resource-constrained settings. In addition, human interaction can help resolve arising ambiguities from data scarcity.

This work contributes several effective and practical robotic perception systems built by combining pretrained models with lightweight modules. The developed interactive preparation procedures demonstrate their strengths in challenging scenarios, particularly for highly similar categories.

Kurzfassung

Diese Dissertation befasst sich mit der Herausforderung, eine objektzentrierte visuelle Wahrnehmung für Roboteranwendungen mit begrenzten Ressourcen zu ermöglichen, dazu gehören beispielsweise begrenzte Trainingsdaten und begrenzte Vorbereitungszeit. Zwar sind vollständig überwachte Deep-Learning-Ansätze qualitativ überlegen, doch können ihre daten- und rechenintensiven Vorbereitungsprozesse den Einsatz in realen Roboterszenarien stark einschränken, insbesondere bei zeitkritischen Aufgaben oder wenn die Bedienenden selbst nicht mit den Methoden des maschinellen Lernens vertraut sind. Um diese Einschränkungen zu überwinden, untersucht diese Arbeit, wie vortrainierte Bildverarbeitungsmodelle als Grundlage für visuelle Perzeptionssysteme dienen können, die nur minimales aufgabenspezifisches Training erfordern.

Der erste Teil dieser Arbeit konzentriert sich auf ressourceneffiziente Wahrnehmungsmethoden, die auf vortrainierten, sogenannten *Backbones* aufbauen. Anstatt auf umfangreiche Trainingsverfahren zu setzen, nutzen die entwickelten Ansätze prototypische Darstellungen und nachgelagerte, leichtgewichtige modulare Erweiterungen, um sich anhand von nur wenigen Beispielen an die jeweiligen Erkennungs- oder Detektionsaufgaben anzupassen. Zudem werden verschiedene Techniken für unterschiedliche Ebenen der Aufgabengranularität untersucht, z. B. durch Inferenzmethoden, welche zusätzliche charakteristische Detailmerkmale berücksichtigen.

Der zweite Teil erweitert die vorgestellten Ansätze um interaktives Lernen, bei dem menschliche Bedienende das Modell anleiten und so dessen Verhalten an die jeweilige Aufgabe anpassen können. Durch die Anpassung der Merkmalsgewichtungen oder die Verschiebung des Fokus des Modells auf wichtige Details kann relevantes Kontextwissen an das Modell weitergegeben werden, ohne dafür zusätzliche Trainingsdaten zu benötigen. Das kann besonders in un spezifizierten Few-Shot-Szenarien vorteilhaft sein, in denen die Aufgabenstellung aufgrund mangelnder Daten mehrdeutig bleibt.

Abschließend werden die vorgeschlagenen Methoden anhand von robotikorientierten Datensätzen und Anwendungsszenarien evaluiert. Die Ergebnisse zeigen, dass vortrainierte Modelle vielfältige Möglichkeiten für Einsatzszenarien bieten, in welchen mit beschränkten Ressourcen ausgekommen werden muss. Darüber hinaus können mittels menschlicher Interaktion Mehrdeutigkeiten bei Datenknappheit beseitigt werden.

Insgesamt präsentiert diese Arbeit mehrere effektive und praktische robotische Wahrnehmungssysteme, die durch die Kombination von vortrainierten Modellen und leichtgewichtigen Zusatzmodulen aufgebaut wurden. Interaktive Einlernverfahren zeigen ihre Stärken insbesondere in Szenarien mit sehr ähnlichen Kategorien.

Statement on AI-Assisted Tools

Generative AI-based tools, including Claude, ChatGPT, DeepL, and Grammarly, were used as supporting aids in the preparation of this dissertation. Their use was limited to language corrections, stylistic and wording suggestions, as well as support in correcting and improving self-written code. All AI-generated suggestions were independently reviewed and are the responsibility of the author.

Contents

Acknowledgments	i
Abstract	iii
Kurzfassung	v
Notation	xi
1 Introduction	1
1.1 Motivation	1
1.2 Problem Statement	2
1.3 Approach Overview	4
1.4 Contributions	5
1.5 Document Outline	6
2 Background	9
2.1 Object Recognition and Detection	9
2.1.1 Object Recognition	9
2.1.2 Object Detection	11
2.1.3 Evaluation Datasets	12
2.1.4 Evaluation Metrics	14
2.2 Model Pretraining	15
2.3 Interpretability Methods	17
2.3.1 Attention in Vision Transformers	18
2.3.2 Concept Activation Vectors (CAV)	19
2.4 General Conclusions	20
3 Related Work	23
3.1 Visual Few-Shot Learning	23
3.1.1 Definitions and Terminology	24
3.1.2 Object Recognition Methods	26
3.1.3 Object Detection Methods	28
3.1.4 Incremental Few-Shot Object Detection	30
3.1.5 Conclusions	31
3.2 Few-Shot Perception Benchmarks	32
3.2.1 Image Classification	32

3.2.2	Object Detection	33
3.2.3	Conclusions	34
3.3	Interactive Learning and Perception	35
3.3.1	Interactive Methods	36
3.3.2	Conclusions	37
3.4	Summary	37
4	Pretraining-based Few-Shot Perception	39
4.1	Frozen Vision Transformers for Attention-Based Segmentation	41
4.1.1	Target Scenarios	41
4.1.2	Architecture and Method	42
4.1.3	Model Preparation	44
4.1.4	Inference and Operation	45
4.2	Transfer Learning with Pretrained Detection Models	46
4.2.1	Target Scenarios	46
4.2.2	Architecture and Method	47
4.2.3	Model Preparation	51
4.2.4	Inference and Operation	52
4.3	Part Prototypes for Fine-Grained Few-Shot Object Recognition with Frozen Models	53
4.3.1	Target Scenarios	54
4.3.2	Architecture and Method	54
4.3.3	Model Preparation	58
4.3.4	Inference	59
4.4	Part-Enriched Prototypes for Fine-Grained Few-Shot Object Detection with Frozen Models	60
4.4.1	Target Scenarios	61
4.4.2	Architecture and Method	61
4.4.3	Model Preparation	66
4.4.4	Inference	67
4.5	Summary and Conclusion	68
5	Interaction-based Few-Shot Perception	71
5.1	Interactive Semantic Modification of Part Prototypes	72
5.1.1	Target Scenarios	73
5.1.2	Architecture and Method	73
5.1.3	Model Preparation	76
5.1.4	Inference	77
5.2	Interactive and Weakness-Aware Prototype Enhancement for Fine-Grained Detection	78
5.2.1	Target Scenarios	78
5.2.2	Architecture and Method	79

5.2.3	Model Preparation	81
5.2.4	Inference	82
5.3	Summary and Conclusion	84
6	Evaluation	85
6.1	Evaluation of Proposed Architectures	86
6.1.1	Frozen Vision Transformer for Attention-Based Segmentation	86
6.1.2	Fine-Tuned Pretrained Models	94
6.1.3	Part-Based Few-Shot Recognition	99
6.1.4	Fine-Grained Few-Shot Detection	109
6.2	Results in Research Applications	118
6.2.1	ROBDEKON	118
6.2.2	ENRICH 2025	120
6.2.3	i-SAMLR	120
6.3	Discussion	122
7	Conclusion and Future Work	125
7.1	Summary	125
7.2	Conclusion	126
7.3	Perspectives for Further Research	128
7.4	Lessons Learned	129
	Own Publications	131
	Supervised Student Theses	133
	Bibliography	135
	List of Figures	153
	List of Tables	155
	Acronyms	157
	Glossary	161
	Appendix	
A	Approach Supplements	163
A.1	LiViT	163
A.1.1	ViT-B Backbone.	163
A.1.2	Mask R-CNN	164
A.2	BatterySort	164
A.2.1	Mask R-CNN	164

A.3	VINTER	165
A.3.1	Backbones	165
A.3.2	Lightweight Modules	165
A.3.3	Concept Activation Vectors	165
A.4	DE-fine-ViT	168
A.4.1	DE-ViT Configuration	168
A.4.2	Incremental Class Addition	168
A.4.3	Fine-Grained Power Drill Dataset	168
B	Robotic Vision Datasets	171

Notation

This chapter summarizes the notation and symbols used throughout this thesis. The notation is kept independent of any specific architecture. Minor deviations from these conventions are used locally where they aid readability or align with established notation in the cited literature.

General Notation

Scalars	Roman and Greek letters	x, α, N
Sets	calligraphic uppercase letters	$\mathcal{D}, \mathcal{C}_{\text{base}}$
Vectors	bold Roman lowercase letters	$\mathbf{x}, \mathbf{f}_{i,j}$
2D, 3D & 4D tensors	bold Roman uppercase letters	$\mathbf{F}, \mathbf{W}$
Random variables	uppercase Roman letters (context disambiguates)	X, Y

Indices i, j usually refer to spatial positions (patch row and column) or set indices. The notation $\|\cdot\|_2$ is used for the Euclidean norm. The symbol $\odot$ denotes the elementwise (Hadamard) product. Matrix slicing usually follows Python conventions, e.g., $\mathbf{X}[a, b:c]$ denotes row a from column b (inclusive) to column c (exclusive).

Few-Shot Learning

N	number of categories (<i>ways</i>) in a few-shot task
K	number of support images per category (<i>shots</i>)
$\mathcal{C}_{\text{base}}, \mathcal{C}_{\text{novel}}$	base and novel class sets
$\mathcal{D}_{\text{base}}, \mathcal{D}_{\text{novel}}$	base and novel datasets
$\mathcal{D}_{\text{support}} = \{(x_i, y_i)\}_{i=1}^{N \cdot K}$	support set for a few-shot task
$\mathcal{Q}$	query set
$f_{\theta} = h \circ g$	model decomposed into backbone g and head h

Image and Patch Representations

$\mathbf{X} \in \mathbb{R}^{H \times W \times 3}$	input image (height H , width W)
P	patch size (pixels)

$H_p \times W_p$	patch grid dimensions ($H_p = H/P$, $W_p = W/P$)
$\mathbf{F} \in \mathbb{R}^{H_p \times W_p \times D_F}$	patch-based image feature tensor
$\mathbf{f}_{i,j} \in \mathbb{R}^{D_F}$	feature vector for patch at position (i,j)
D	embedding dimension of the backbone (single layer)
$\mathbf{F}_O \in \mathbb{R}^{H_p \times W_p \times C_O}$	output-layer patch features (C_O -dimensional)

Backbone Internals

The symbols below apply to layered backbones in general and attention-related entries specifically apply to vision transformers.

ℓ	layer index
h	attention head index
N_{heads}	number of attention heads per layer
$N_{\text{tkn}} = H_p \cdot W_p$	number of patch tokens
$\mathbf{Q}^{(\ell,h)}, \mathbf{K}^{(\ell,h)}, \mathbf{V}^{(\ell,h)}$	query, key, value matrices at layer ℓ , head h
$d_k = D/N_{\text{heads}}$	key dimension per head
$\mathbf{A}^{(\ell,h)}$	attention weight matrix at layer ℓ , head h
$\bar{\mathbf{A}}^{(\ell)}$	head-averaged attention at layer ℓ
$\mathbf{R}^{(\ell)}$	attention rollout matrix at layer ℓ

Part-Based Recognition

N_p	number of parts (clusters)
$\hat{\mathbf{M}}_{\text{fg}} \in \mathbb{R}^{H_p \times W_p}$	raw foreground score map
$\mathbf{M}_{\text{fg}} \in \mathbb{R}^{H_p \times W_p}$	clipped foreground map, values in $[0,1]$
$\hat{\mathbf{M}}_{\text{part}} \in \mathbb{R}^{H_p \times W_p \times N_p}$	raw part segmentation map
$\mathbf{M}_{\text{part}} \in \mathbb{R}^{H_p \times W_p \times N_p}$	foreground-masked part segmentation map
$\hat{\mathbf{M}}_{\text{sim}} \in \mathbb{R}^{H_p \times W_p \times N_p}$	raw part similarity map
$\mathbf{M}_{\text{sim}} \in \mathbb{R}^{H_p \times W_p \times N_p}$	clipped part similarity map
$\tilde{\mathbf{M}}_{\text{sim}}$	NMS-filtered part similarity map
$\hat{s}_p \in [0,1]$	aggregated similarity score for part p
$\hat{\mathbf{s}} = [\hat{s}_1; \dots; \hat{s}_{N_p}]^T \in \mathbb{R}^{N_p}$	part score vector
w_p	operator-adjustable weight for part p

Class and Part Prototypes

$\mathbf{p}_{c,i}^{(\text{cls})}$	per-image class prototype for category c , support image i
$\mathbf{p}_c^{(\text{cls})}$	global class prototype for category c
$\mathbf{p}_{c,j}^{(\text{part})}$	part prototype j for category c
$\mathbf{P}_{\text{cls}} \in \mathbb{R}^{N \times D}$	class prototype bank (N categories)
$\mathbf{P}_{\text{bg}} \in \mathbb{R}^{N_{\text{bg}} \times D}$	background prototype bank (N_{bg} prototypes)
$\alpha_{c,j}^{(\text{tkn})} \in [0,1]$	patch mask weight for class c , patch j
$\text{top-}k_{\text{det}}$	number of top candidate classes forwarded to the detection head
$\text{top-}k_{\text{RoI}}$	number of top candidate classes per RoI forwarded to the part-based re-evaluation
$\mathcal{T}_i^{(k)}$	top- k candidate class set for proposal i

Concepts and Interaction

a	semantic concept label (e.g., <i>orange</i> , <i>striped</i>)
$\mathcal{A} = \{a_1, \dots, a_J\}$	concept set (selected subset of available concepts)
$\mathcal{B} = \{\mathbf{v}_{a_j}\}_{j=1}^J$	concept bank (collection of CAVs)
$\mathbf{v}_a^{(\ell)} \in \mathbb{R}^D$	concept activation vector (CAV) for concept a at layer ℓ
$f_{a_j} : \mathbb{R}^D \rightarrow [0,1]$	linear concept predictor for concept a_j
$\alpha_{i,j,p} = \mathbf{w}_p^{\text{pse}} \cdot \mathbf{v}_{a_j}$	concept sensitivity of part p to concept a_j
$\mathcal{V}_i^{\mathcal{A}} = \text{span}(\mathbf{v}_{a_1}, \dots, \mathbf{v}_{a_J})$	concept subspace in the embedding space

Metrics and Thresholds

TP, FP, TN, FN	true/false positives and negatives
$\text{Precision} = TP/(TP + FP)$	fraction of retrieved items that are relevant
$\text{Recall} = TP/(TP + FN)$	fraction of relevant items that are retrieved
$F_1 = 2 \cdot \text{Prec} \cdot \text{Rec} / (\text{Prec} + \text{Rec})$	harmonic mean of precision and recall
AUROC	area under the ROC curve
$\text{IoU}(A, B) = A \cap B / A \cup B $	intersection over union for region A and B
AP	average precision (area under precision-recall curve)
mAP	mean AP over all categories
τ_{conf}	confidence score threshold

τ_{seg}	segmentation mask threshold
τ_{IoU}	IoU threshold for TP/FP assignment

1 Introduction and Overview

This work investigates methods for enabling object-centric visual perception across diverse robotic applications without requiring extensive training data, powerful compute resources, specialized expertise, or time-consuming preparation steps. While recent learning-based vision methods have achieved remarkable performance on diverse benchmarks using large datasets for training, deploying these models in practical robotic scenarios and adapting them to new tasks remains challenging, as resources such as data and time are often too constrained. In contrast, research in the area of few-shot learning moves in the opposite direction. It aims to develop alternative solutions that are applicable when data is scarce. Building on these techniques, extending them, and developing new strategies with the overall goal of obtaining model preparation procedures for stationary and mobile robotic use cases is the core of this dissertation. Example use cases include sorting tasks on conveyor belts or object-retrieval missions with mobile robots. The difference here is that the proposed procedures work with reduced requirements for the main resources, such as the amount of training data, compute hardware, technical know-how, or overall preparation time, compared to common deep learning approaches.

This first chapter introduces the underlying motivation and presents the main research questions and the general approach concept at a higher level. First, [Section 1.1](#) presents general observations that motivated this research, highlighting why common perception models can be overly demanding in applied robotic scenarios. [Section 1.2](#) formally states the research problem and splits it into separate research questions. [Section 1.3](#) provides an overview of the two complementary strategies pursued in this dissertation: resource-efficient methods that minimize requirements during model preparation, and interactive procedures that enable operator guidance without any extensive retraining. Finally, [Section 1.4](#) summarizes the key contributions of this work. [Section 1.5](#) outlines the structure and contents of the upcoming chapters.

1.1 Motivation

The deployment of robotic systems in real-world applications has accelerated dramatically in recent years. Moving beyond controlled factory settings into complex, unstructured environments demands new visual perception capabilities, as the robot will encounter unfamiliar and potentially unexpected objects.

Visual perception in this work refers to tasks involving RGB-based object-centric recognition within the robot's environment, yielding categorical and positional information in the image

as bounding boxes or pixel-level masks. This information can then be used either to react to objects or to interact with them.

In recent decades, remarkable advances in computer vision have been achieved through learning-based approaches for object recognition, surpassing conventionally crafted algorithms and filters. By design, machine learning usually aims to process and learn from as much data as possible. As a consequence, preparing effective perception models for robotic missions remains resource-intensive and time-consuming, and often requires specialized expertise. One reason is that these models typically rely on extensive labeled datasets containing hundreds or even more examples per object category.

When robots are deployed to perceive application-specific objects, collecting and annotating thousands of images becomes a significant bottleneck. Using existing datasets for training to avoid this labor is often not an option, because at least one task-relevant object is typically not part of any accessible dataset. Since creating such datasets for every new application is impractical, this becomes challenging for real robotic scenarios.

Beyond data requirements, the computational resources needed to train deep neural networks present another significant hurdle. State-of-the-art vision models often require powerful Graphics Processing Units (GPUs) and substantial training time, measured in hours or days. Fine-tuning pretrained models can reduce training time and computational requirements, but creating a dataset is still necessary.

In field robotics, particularly in mobile-robot or edge-deployment scenarios, computational resources are constrained to locally available components. If the robot operator has access only to the robot's onboard hardware, the available processing power is considerably lower than on dedicated training workstations or cloud computing infrastructure. Even fine-tuning pretrained models may exceed the capabilities of the available hardware.

The focus of this work lies on robotic use cases that potentially involve very specific or rare items, limited compute hardware, unforeseeable object types relevant to the corresponding mission, or settings in which a short preparation time is crucial. Scenarios with ordinary widespread objects such as cups or chairs, e.g., in common household tasks, are therefore not the primary focus. This is mainly because such items are widely represented in many datasets, and perception models pretrained on them are publicly available. These scenarios do, however, become relevant again when fine-grained recognition of these objects is necessary.

1.2 Problem Statement

Despite ongoing developments in computer vision and deep learning, deploying visual perception systems in robotic applications remains laborious and challenging due to resource limitations. Current state-of-the-art perception techniques require large amounts of labeled training data, expensive computational resources for training or inference, and specialized

knowledge of deep learning and computer vision. As a result, the overall model preparation requires a considerable amount of time. These demands do not align with the realities of deploying perception models for robots, where mission preparation time can be extremely constrained, computational resources are limited to onboard hardware, operators are usually not machine learning experts, or the objects of interest are highly specific.

Few-shot and low-data adaptation methods appear promising for addressing these limitations, as they enable model preparation with only a few examples. Nonetheless, the process of adapting such techniques to the robotic domain and its typical use cases needs to be examined. Here, perception refers to a wide range of visual tasks, including classification, detection, and segmentation, with varying degrees of granularity.

This dissertation examines three research questions related to facilitating a less demanding setup of object-centric visual perception in robot applications:

Research Question 1

How can general-purpose pretrained models be adapted to object-centric robotic perception while reducing preparation requirements?



This question examines whether approaches based on generally pretrained vision models can address the resource constraints that limit the practical deployment of perception systems in robotics. It requires assessing lightweight adaptation strategies that leverage pretrained backbones and evaluating their effectiveness in robotic scenarios. The goal is to investigate how preparation requirements, such as data, compute, time, and expertise, can be reduced, and under what conditions such methods reach a level of performance sufficient for practical use.

Research Question 2

To what extent can few-shot architectures support object-centric perception on different levels of task granularity?



This question explores the applicability of few-shot architectures to robotic perception tasks involving categories at different granularity levels. It investigates how much task-specific adaptation is possible with limited or no fine-tuning, and identifies designs that efficiently support these levels. A particular aspect of this research direction is the development of components that can handle generic and fine-grained categories at the same time.

Research Question 3

How can operator knowledge be integrated through user interfaces to improve task alignment without additional labeled training data?



This question addresses the challenge of enabling non-expert operators to improve perception model performance through interaction. It examines how the operator's contextual knowledge

can be leveraged and translated into model adaptations, and how interactive feedback can compensate for limited labeled training data while remaining as accessible as possible to users without in-depth machine learning expertise.

1.3 Approach Overview

Today, data has become one of the most important factors for perception methods based on machine learning. The usual paradigm is: the more data, the better the model. However, in robotic applications, limitations are common. These concern not only the data available for the task at hand, but also other resources such as compute, connectivity, power, or time. For instance, if robots are deployed to time-critical tasks involving unforeseeable objects, common strategies for setting up learning-based perception systems are not feasible.

This conflict is the subject of the research presented in this work. The goal is to enable practically usable perception systems across various types of robotic applications despite these limitations. To this end, task-specific training efforts should be minimized and large-scale data collection avoided.

The core principles of the presented methods revolve around pretrained models and how they can be incorporated into specific perception tasks. These models have already processed large amounts of data during pretraining. However, the training data is unlikely to cover all objects encountered in robotic applications. Nevertheless, their developed capabilities for understanding the *visual world* are leveraged to solve new, specific tasks with minimal adaptation. While this idea is not new, this work presents the concepts of the developed perception systems, including assistive tools, specialized architecture modules, and interactive interfaces for the human operator of the robot. The tools support data collection and annotation, while the architecture modules are built on top of generally pretrained models to leverage their representations and capabilities. Across the proposed methods, these representations are processed, structured, and redirected to solve generic recognition tasks, and dedicated variants further handle the fine-grained differentiation of similar categories for which only little data is available. However, when the data defining the categories is scarce, distinguishing causal features from spurious correlations becomes more difficult, making underspecification and shortcut learning more likely. This is where the interactive interfaces come into play. Their purpose is to translate operator knowledge into the model's domain, so that the model can benefit from it and focus on truly relevant features rather than on those that appear important based on the few samples available.

The overarching objective is to reduce the requirements across the three phases of setting up a perception model by enabling systems that can operate on only a few data samples. These phases are data collection and annotation, model training and evaluation, and model integration and inference on the robotic system (Fig. 1.1). By reducing the demand for large amounts of training data, avoiding the need to extensively optimize model parameters, and enabling a more direct form of human input and feedback, the preparation phase becomes more compact,

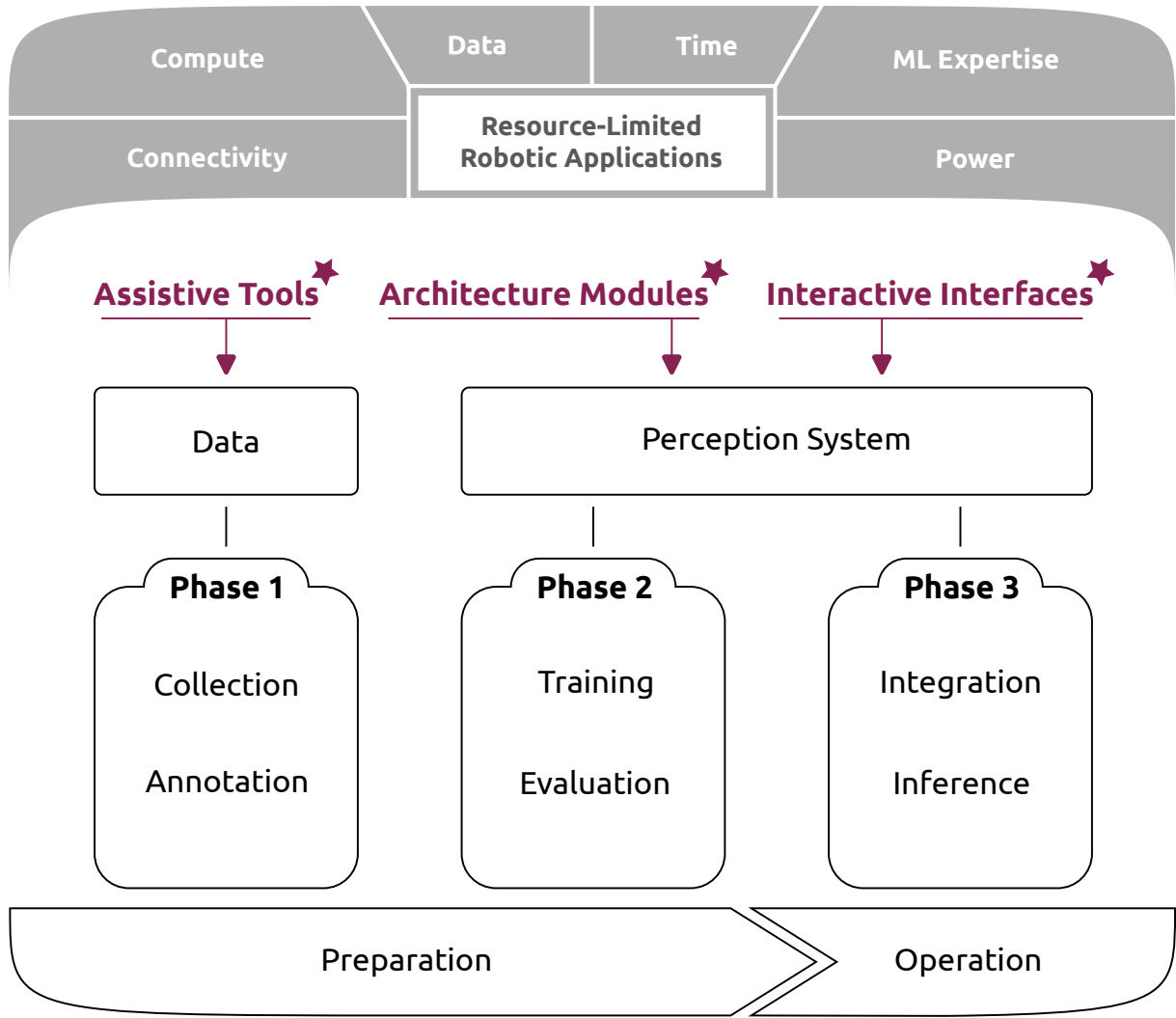


Figure 1.1: High-level overview of the concept developed in this work. The main research area is the preparation and operation of perception systems in the context of resource-limited robotic applications. Novel architectures are proposed that integrate modules designed for few-shot learning, which provide optional interactive interfaces for human feedback. Assistive tools for data handling are incorporated to reduce the operator’s workload.

which improves usability in resource-limited robotic applications. Nonetheless, robustness and performance cannot be disregarded in favor of resource savings. These properties often determine whether a system is usable in practical, real-world scenarios. Therefore, this work examines how the proposed methods weigh limited data against robustness, and where that trade-off breaks down.

1.4 Contributions

This section provides an overview of the contributions presented in this work. In-depth details of the contributions are elaborated in the corresponding chapters. The core contributions are:

- ◇ Leveraging pretrained models and developing architectures that *stand on their shoulders* to support few-shot learning for various object-centric perception tasks in robotic applications. Instead of creating large datasets for the task at hand, possibilities are explored to use significantly smaller datasets and to leverage the inherent capabilities of the pretrained models as much as possible. For instance, an approach for segmenting litter items in ground images uses pretrained models with their parameters kept completely frozen. It leverages the model’s attention learned during pretraining on a different task to obtain segmentation masks. In another scenario, where the samples of a single category differ strongly from each other, a fine-tuning procedure for the higher layers is supported by incorporating additional 3D depth data to create a hybrid prediction pipeline. When categories are almost identical and only subtle visual details set them apart, part-based inference methods are proposed for classification and detection tasks.
- ◇ Providing interfaces that allow the operator to better understand the model’s internal perception, as well as interfaces that *translate* the operator’s context knowledge into the model’s *language* to improve the task alignment. Usually, when a trained model produces incorrect predictions on test samples, the typical remedies are collecting more data or adjusting training hyperparameters. To investigate a third option, special model architectures are equipped with interfaces that allow the operator to directly adjust the model’s focus on relevant parts. This approach can be beneficial in cases where two categories are mostly identical except for a few subtle differences that are hard for the model to infer from the few samples provided.

Although not a core contribution, during this dissertation many existing datasets related to perception in robotic applications were reviewed. The collected list is publicly available online and briefly described in [Chapter B](#) of the appendix. The included datasets can be used for evaluating state-of-the-art perception systems. While the list also contains many popular benchmarks, the goal was to integrate lesser-known datasets with less common and rare items that, for instance, could serve as a source of annotated training or validation data.

1.5 Document Outline

This chapter has covered the motivation and the formulation of the research questions. It has further described the contributions and provided a high-level overview of the proposed approach. The remainder of this work is structured as follows:

CHAPTER 2 This chapter introduces the methodological basis for the presented state of the art and the proposed approaches. Its purpose is to recap and explain the most relevant concepts required for the comprehension of all subsequent chapters.

CHAPTER 3 Presents the related work in the domain of computer vision for robotics, with a particular focus on few-shot learning and corresponding benchmarks. State-of-the-art methods for human interaction with perception models are also discussed. The latter is specifically relevant as a reference for [Chapter 5](#).

CHAPTER 4[★] Derives and describes the proposed methods for leveraging pretrained models in robotic perception. Several scenarios are addressed using specialized architectures with the common objective of reducing dependence on large datasets for training. The architectures rely entirely on data for task preparation. The next chapter extends these architectures with interactive interfaces for the operator.

CHAPTER 5[★] Derives and describes the proposed interactive model preparation strategies for robotic perception. The methods build on the architectures presented in the previous chapter and aim to incorporate the operator into the model’s task preparation. When data is limited, human context knowledge becomes valuable. The interfaces implicitly allow the exchange of such additional information.

CHAPTER 6[★] This chapter covers the qualitative and quantitative evaluation of the methods introduced in [Chapters 4](#) and [5](#). It includes practical experiments and real-world scenarios. Results and findings are critically discussed at the end of the chapter.

CHAPTER 7 Summarizes and concludes this work. Future work is motivated by the current state of research. A short, informal review of the lessons learned is appended as well.

2 Background

This chapter introduces the research areas and the most common methods relevant to this work. The goal is to derive current standards in image-based perception from a historical perspective. In this work, perception refers to 2D RGB images unless stated otherwise.

This chapter shows how steady improvements in image classification and detection led to modern object architectures, which often form the foundation for few-shot methods ([Chapter 3](#)). It also examines the role of model pretraining in computer vision and which techniques are used for which purposes. Finally, different attempts at interpretability of artificial neural networks are briefly explained together with a short overview of the challenges they entail.

2.1 Object Recognition and Detection

This section briefly explains the scientific evolution of methods for object recognition and detection in images. This is often not only interesting for the engaged reader but also helpful for understanding current approaches, since it shows when and why the corresponding components were developed and proposed. The difference between recognition and detection is that recognition architectures predict the probability of an image belonging to a certain category, while detection architectures not only estimate the category of multiple objects but also hypothesize their locations within the image. Usually, this hypothesis contains positional information defined by a bounding box aligned with the image coordinate system. For reasons of relevance, the following sections focus on methods based on artificial neural networks, and the list is by no means exhaustive.

2.1.1 Object Recognition

One of the first popular artificial neural network architectures for object recognition is the 7-layer LeNet-5 by LeCun *et al.* [1]. It was initially developed to recognize handwritten characters (zip codes). The foundations for this network were laid by LeCun *et al.* [2], who showed that a network with 2 convolutional layers followed by 2 fully connected layers can take images directly as input (after simple preprocessing) and can be trained end-to-end with the backpropagation algorithm [3]. This removed the need for sophisticated, hand-crafted image processing modules for feature extraction or classification, which were common at that time. Another advantage is that it integrates convolutional network modules, which have built-in architectural inductive biases that encode *a priori* knowledge about 2D images and ultimately reduce the number of parameters. For instance, they inherently leverage the locality of visual information compared to fully connected network layers. Convolutions, defined by kernels

(multi-dimensional matrices), only use information from the local neighborhood at a given position when extracting object features (characteristic patterns). Furthermore, convolutional layers exhibit layer-level translation equivariance and aim for system-level translation invariance. Thus, the object's intermediate features stay the same but are translated if the object is moved within the image, and the prediction remains the same regardless of the object's location in the input image.

In the popular ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [4], which is based on the ImageNet dataset [5], AlexNet [6] was the first Convolutional Neural Network (CNN) to win the competition in 2012, starting an era of CNN dominance in this challenge. The challenge was first held in 2010 and continued annually until 2017. In the first two years, methods based on support vector machines and bag-of-visual-words took the lead, although CNNs were already under active research. AlexNet was the first to change this. Proposed by Krizhevsky *et al.* [6], it has 5 convolutional and 3 fully-connected layers. The authors leveraged Graphics Processing Unit (GPU) execution of CNN-related functions and applied techniques to avoid overfitting to better handle the large amounts of data in this challenge. They also showed that activation functions matter for training speed and that rectifying linear units (ReLUs) $f(x) = \max(0, x)$ require significantly fewer training epochs than, for example, the hyperbolic tangent $f(x) = \tanh(x)$, suggesting that this correlates with the non-saturating character of ReLUs. The term rectifying is borrowed from electrical engineering, where it refers to removing the negative half of an alternating voltage. Following AlexNet's breakthrough, several architectural innovations further improved recognition accuracy in the ILSVRC. While it was common to create deeper networks, e.g., the Visual Geometry Group (VGG) models [7] with up to 19 layers (VGG-19), bottlenecks on the hardware side and in learning algorithms had to be addressed step by step. By stacking multiple 3×3 convolutional filters rather than using larger kernels as previously done, the VGG architecture achieved an effective receptive field equivalent to that of larger filters while reducing the number of parameters. Batch normalization, introduced by Ioffe and Szegedy [8], addressed the problem that the input distribution of a layer depends on the output of the previous layer and changes constantly during training. They called this internal covariate shift and fixed it by normalizing layer inputs, which allows higher learning rates and greater robustness to network parameter initialization. Residual Networks (ResNets) by He *et al.* [9] solved the error signal degradation problem arising in deep networks through skip connections that enable identity mappings. This trick enabled effective training of networks with hundreds or thousands of layers and has been widely adopted by other architectures. ResNets of various depths, e.g., ResNet-18, ResNet-50, or ResNet-101, have become one of the default backbones and remain so today. Despite the large number of layers, the complexity of a 152-layer ResNet (11.3 billion FLOPS) is lower than that of a VGG-16 (15.3 billion FLOPS) [9]. The maximum depth during evaluation was 1202 layers. Interestingly, the optimization during training succeeded, but the test scores on the CIFAR-10 [10] dataset were worse than those of the ResNet with 110 layers.

With the transformer architecture [11], a purely attention-based model, the field of language models had its *AlexNet moment*, and formerly popular recurrent networks were replaced.

Dosovitskiy *et al.* [12] transferred the success of these attention mechanisms to image processing and classification by introducing the Vision Transformer (ViT). Several promising attempts by other researchers preceded this successful transfer. ViTs lack some of the inductive biases that CNNs have by design, and on ImageNet their performance is slightly inferior to that of a ResNet of similar complexity. However, when pretrained on larger datasets with more than 14M images, their real strengths become evident. Initializing training on smaller datasets with weights from large-scale pretrained ViTs yields new state-of-the-art results on several image recognition benchmarks.

Swin Transformers [13] propose a hybrid transformer-based architecture with convolutional elements, e.g., locally constrained self-attention similar to convolutional filters, and hierarchical feature processing with decreasing resolution while the embedding space dimensions grow in higher layers, which enables denser downstream tasks such as detection. They achieve competitive performance on image classification benchmarks and even beat comparable ViTs and ResNets when trained on the same data. In contrast, ConvNeXt models [14] return entirely to convolutional network modules and show that careful optimization can outperform ViTs and Swin Transformers without any attention mechanisms.

2.1.2 Object Detection

The task of object detection is to predict the categories of objects in an image as well as their positions in image coordinates. Object positions are most commonly defined by rectangular bounding boxes aligned with the image frame. At a higher level, object detection methods can be divided into two categories. Where-first approaches, also called two-stage models, first identify likely object locations and then refine the positional information while classifying the detected content. One-stage models, in contrast, determine location and category simultaneously, answering where and what in a single pass. In the following, representative methods and architectures for both types are presented in detail, followed by a look at very recent architectures and concepts for object detection.

Generally speaking, two-stage or position-first methods perform image classification on smaller image regions, often combined with a regression sub-task that refines the initial position estimates. A popular example is Region Proposal Networks (RPNs) introduced by Ren *et al.* [15]. A RPN is a fully-convolutional network designed to share the detection backbone in order to reduce computational overhead. It predicts multiple proposals at each position in the input feature map, i.e., the shared backbone's output features. A proposal consists of the region coordinates and an objectness score. It can therefore be seen as class-agnostic, although it naturally has a bias towards objects contained in its training dataset. The coordinates are predicted as regression offsets relative to one of multiple anchor box variants at each feature map position. RPN training follows specialized procedures, either alternating with the detection network or in joint optimization with it. One well-known predecessor of RPNs for estimating class-agnostic object locations in images was Selective Search [16], which is based on traditional image processing methods. It usually ran on the Central Processing

Unit (CPU) rather than the GPU, which made it slow. Still, it was one of the most effective methods before RPNs. For instance, it was used in the Region-based Convolutional Neural Network (R-CNN) architecture [17], a predecessor of the popular Faster R-CNN [15]. R-CNN, however, processed each region separately for feature extraction. Fast R-CNN [18] showed that processing the whole image once and then performing a pooling operation over those features for each proposal is much more efficient and brings several other advantages. The Region of Interest (RoI) pooling works by converting the image features corresponding to the bounding box into a representation with fixed dimensions independent of the actual box size, e.g., 7×7 . It splits the original proposal into a corresponding number of subregions and then applies simple maximum pooling in each subregion. R-CNN already used a similar pooling strategy, but instead of pooling extracted features, it warped the image regions into a fixed-size input for the CNN. With efficient feature processing and alignment, the new bottleneck became the generation of region proposals. As mentioned above, this was solved by the Faster R-CNN model using RPNs.

Unlike two-stage methods, one-stage methods directly predict class probabilities and bounding box coordinates in a single network pass. Well-known candidates include Single Shot Multi-Box Detector (SSD) [19] and You Only Look Once (YOLO) [20], for which updated architecture variants remain extremely popular. YOLO divides images into a grid and simultaneously predicts bounding boxes and class probabilities for each grid cell. The unified architecture enabled real-time detection speeds, but initially struggled with small objects and precise localization.

Recent developments move away from predefined anchor boxes, e.g., by formulating the detection problem as corner prediction (CornerNet [21]) or center prediction (CenterNet [22]). Likewise, Fully Convolutional One-Stage object detector (FCOS) [23] is fully anchor-free and proposal-free. Instead, for each position in feature maps at various depths, they predict a classification score (center-ness) and a box regression.

A new hybrid architecture emerged with DETection TRansformer (DETR), proposed by Carion *et al.* [24]. They reformulated object detection as a set prediction problem and used transformers that operate on top of CNN-based feature extractors. By employing a bipartite matching loss and learnable object queries, DETR removed the need for several hand-designed components, such as anchor generation and Non-Maximum Suppression (NMS).

2.1.3 Evaluation Datasets

This subsection describes common datasets used for benchmarking object recognition and object detection approaches. In fully supervised learning, the number of training samples is not restricted and is preferably as high as possible. In contrast, typical few-shot settings allow 1-30 samples per category during the model adaptation phase. Therefore, few-shot benchmarks (Section 3.2) usually adapt a dataset originally designed for fully supervised training and define precise procedures for obtaining task-specific support sets and query set images.

Early larger recognition datasets covered digits, e.g., MNIST [1]. With CIFAR-10 [10], which contains 10 classes of different animals and vehicles, and CIFAR-100 [10], which contains 100 subclasses each belonging to one of 20 superclasses, more natural object types were covered: household items, plants, animals, vehicles, people, and various scenes. However, the images have a relatively low resolution of 32×32 pixels. ImageNet [5] was one of the first public datasets to provide images at moderate resolution in large quantities. The ImageNet-1k version covers 1000 classes, while ImageNet-21k (also referred to as ImageNet-22k) covers almost 22,000 classes. The smaller version is a subset of the larger one. Both are still used as the main benchmark for state-of-the-art image classification models. Beyond that, they are common pretraining datasets for a wide range of vision architectures, especially for integrated backbones. For fine-grained recognition, in addition to CIFAR-100, there is the popular iNaturalist dataset [25], which comprises more than 5000 plant and animal species, and the CUB-200-2011 dataset [26], which covers 200 bird species.

For object detection, the PASCAL VOC challenge provided an annually updated dataset [27] with up to 20 common classes, ranging from airplanes to TVs. The goal was to obtain natural images with high variability in object size, background, and lighting conditions. There are two common versions: PASCAL VOC 2007, the first year with 20 classes and pixel-level segmentations, and PASCAL VOC 2012, the last and most popular version. The dataset comes with clearly defined train, val, and test splits [27] aimed at standardized evaluation.

Another benchmark for object detection is the MS COCO dataset [28], or simply COCO. Its annotations comprise 80 publicly available categories, whereas the original publication mentions 91. The goal was to create a dataset for object detection with items that are “easily recognizable by a 4-year-old” [28, p. 740]. In general, it is quite similar to the PASCAL VOC datasets, but at a significantly larger scale. Instead of around 10k images with approximately 25k instances, COCO contains around 300k images with 2.5 million annotated instances obtained via crowdsourcing. The 20 PASCAL VOC classes are also part of the 80 COCO classes. There are two dataset versions: COCO 2014 and COCO 2017. While the images and annotations largely remained the same, the split into training and validation sets was adjusted in the later version to improve standardized evaluation. The training split grew from 82,783 to 118,287 images, and the validation split was reduced from 40,504 to 5,000. All 5,000 of these images had been part of the COCO 2014 val split, also referred to as COCO *val2014*.

The next-largest dataset for object detection is Objects365 [29], with 365 categories, more than 600,000 images, and around 10M bounding boxes. The authors showed that it is superior to ImageNet-1k for pretraining object detection architectures. The 80 COCO categories are also part of Objects365. The dataset contains images at higher resolutions than COCO.

Another frequently used dataset is LVIS [30]. Its pixel-level annotations are used not only for object detection but also for instance segmentation. The number of masks, more than 2M in total, is intentionally unbalanced across the more than 1.2k categories. The rationale is that object categories in the real world also do not appear equally often, which makes the dataset suitable for more realistic benchmark scenarios. Note that some categories have fewer than

10 labeled instances, which makes them challenging to use for training and evaluation. All of the around 164k images are part of the COCO 2017 dataset.

2.1.4 Evaluation Metrics

This section describes the most commonly used evaluation metrics for classification, detection, and segmentation tasks. This is not a complete list but rather a selection of the most relevant ones for this work. In the following, TP is short for the number of *True Positives* and FN for the number of *False Negatives*. Accordingly, FP denotes the number of *False Positives* and TN the number of *True Negatives*.

- **Accuracy:** Describes the overall correctness of the model's predictions, but must be handled with caution if positive and negative samples are not well balanced.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.1)$$

- **Precision & Recall:** Precision measures prediction reliability, i.e., the ratio of correct positive predictions to all positive predictions. Recall, in contrast, measures how completely all true positives are retrieved. The higher the recall, the fewer positive samples are falsely classified as negative and are *overlooked*. Recall is also known as True Positive Rate (TPR) or *sensitivity*.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2.2)$$

$$\text{Recall (= TPR)} = \frac{TP}{TP + FN} \quad (2.3)$$

- **F1 Score:** The harmonic mean of precision and recall. The motivation is to combine precision and recall into a single metric. Optimizing only for recall or only for precision is often not desirable. Since the score does not include TN , it is suitable even when negative samples are far more numerous. Note that a similar metric for combining TPR and False Positive Rate (FPR) exists, called Youden's index.

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2TP}{2TP + FP + FN} \quad (2.4)$$

- **Sensitivity, Specificity & FPR:** While sensitivity, also known as TPR or recall, measures the share of correctly retrieved positive samples, specificity inversely evaluates the correctly recognized negative samples. Derived from specificity, the FPR

provides a measure of the ratio of *false alarms*, e.g., when they are costly.

$$\text{Sensitivity (= TPR = Recall)} = \frac{TP}{TP + FN} \quad (2.5)$$

$$\text{Specificity} = \frac{TN}{TN + FP} \quad (2.6)$$

$$\text{FPR} = \frac{FP}{FP + TN} = 1 - \text{Specificity} \quad (2.7)$$

- **Area Under the Receiver Operating Characteristic curve (AUROC):** Provides a threshold-independent measure of predictor quality. It refers to the area under the receiver operator characteristic (ROC) curve. The Receiver Operating Characteristic (ROC) space is spanned by the TPR (y-axis) and the FPR (x-axis). Similar to the ROC space, the precision-recall (PR) space plots precision against recall. However, comparing binary predictors in these two spaces can make a significant difference [31]. For sample data with $|\text{positives}| \ll |\text{negatives}|$, the AUROC score can appear overly optimistic.

$$\text{AUROC} = \int_0^1 \text{TPR}(\tau) d(\text{FPR}(\tau)) \quad (2.8)$$

- **Intersection over Union (IoU):** A core localization metric for bounding boxes and segmentation masks. An IoU threshold is often used to determine whether a predicted image area counts as a *TP* or *FP*.

$$\text{IoU} = \frac{|b_{\text{pred}} \cap b_{\text{gt}}|}{|b_{\text{pred}} \cup b_{\text{gt}}|} \quad (2.9)$$

- **Average Precision (AP):** The area under the precision-recall curve. It is robust for imbalanced detection tasks. The mean Average Precision (mAP) stands for mean AP across all categories. In few-shot learning, nAP and bAP are also common, denoting the mean AP across novel and base categories, respectively.

$$\text{AP} = \int_0^1 \text{Precision}(r) dr \quad (2.10)$$

2.2 Model Pretraining

Pretraining refers to an initial training phase on a dataset, distinct from the downstream target task. The learned parameters serve as the initialization for subsequent task-specific training. One of the earliest appearances of this practice is the method proposed by Donahue *et al.* [32], which analyzes in detail whether trained convolutional feature extractors can be reused for new recognition tasks, even in other domains that lack sufficient training data. In the same

year, Girshick *et al.* [17] showed that pretraining a backbone for image classification on ImageNet benefits their proposed object detection architecture (R-CNN) on the PASCAL VOC dataset. Note that the backbone serves as a region proposal classifier in the R-CNN model. Also in 2014, Yosinski *et al.* [33] studied layer-wise characteristics of transferability. They showed that early layers learn more general and universal features, while deeper layers become increasingly task-specific.

Pretraining a vision model before fine-tuning it on the actual task is a form of transfer learning. Transfer learning broadly describes knowledge transfer from a familiar source task to a new target task [34] and, in the vision domain, involves many more decisions on top of the pretraining and fine-tuning steps.

Closely related is the term representation learning, which aims to learn embeddings useful for the desired downstream tasks. A common tool to evaluate the *quality* of backbone representations is linear probing, which attaches a simple classification layer, typically a linear one, to the output of the corresponding model layer [35]. With the backbone kept frozen, only the probe is trained for classification. The final performance of the probe improves as the representations get better.

Self-Supervised Representation Learning

A recent self-supervised learning strategy called self-Distillation with NO labels (DINO) [36] applied to vision transformers showed that the resulting features carry stronger semantic information about objects in the image than those obtained from supervised training or convolutional networks. The attention maps contain information about object boundaries and parts, even though no segmentation supervision was used during training. The pretraining was conducted on the ImageNet dataset [4] without labels and combined with multiple data augmentation strategies. These features can be used for various downstream tasks such as pose estimation [37] or co-part segmentation [38]. The co-part segmentation technique was also used for evaluation comparisons in a one-shot affordance transfer approach [39], although it was not the best-performing one. With an updated self-supervised training recipe, DINOv2 [40] scaled up the data and model size, yielding even *better* features. The resulting models produce features that transfer competitively with, or exceed, supervised pretraining across diverse benchmarks, including classification, segmentation, and depth estimation. They trained a 1.1B ViT-G model and used knowledge distillation [41] for a series of smaller ViT models. The pretraining data is the LVD-142M dataset, a curated collection without image or object labels. Multiple splits of common datasets served as image retrieval anchors, including the train split of CUB-200-2011 and ImageNet-1k. Using images from these reference datasets, similar images were retrieved from a large, uncurated pool using cluster-based methods. However, ImageNet-21k was used for an initial phase of pretraining, and ImageNet-1k is part of it. While the authors explicitly state that they performed copy detection and removal for images in any validation or test set of the datasets they used, their descriptions suggest that the train split of ImageNet-1k is included in LVD-142M.

Subsequent studies revealed high-norm tokens, so-called artifacts, in the attention maps of DINOv2 and other modern vision transformers, especially in image regions with little local content. To avoid such high-norm tokens, Darcet *et al.* [42] proposed adding special tokens dedicated to this purpose. These tokens are called register tokens. Interestingly, in its first version, DINO [36] was the only ViT not exhibiting such artifacts.

The mentioned ViTs from the DINO family, pretrained without supervision, have seen large amounts of unlabeled but curated data. They achieve performance competitive with fully supervised models on classification benchmarks, both with linear classification layers on top and even when using only k-NN prediction.

Supervised Representation Learning

Supervised pretraining, on the other hand, uses large labeled datasets to learn transferable features. Supervised ImageNet pretraining is a common starting point for downstream tasks. The target domain should share visual characteristics with the natural images contained in ImageNet. Over time, larger datasets have been released and adapted for supervised representation learning. Contrastive Language-Image Pre-training (CLIP), proposed by Radford *et al.* [43], took a slightly different approach. Instead of training on images alone, they use a dataset of around 400 million image-text pairs collected from the internet and apply contrastive learning losses between image and text embeddings while jointly training an image encoder and a text encoder. As a result, the image encoder’s representations naturally support zero-shot learning-style classification via natural language prompts and also transfer well to diverse visual tasks. The Bootstrapping Language-Image Pre-training v2 (BLIP-2) [44] framework goes one step further. While the framework itself addresses vision-language pretraining of large-scale models, the vision encoders it selects are relevant for this work. It uses completely frozen image encoders, which are later combined with language models via a lightweight querying transformer. The frozen image encoders chosen are CLIP [43] and the larger variant EVA-CLIP [45]. EVA on its own is a 1B-parameter vision transformer pretrained in a self-supervised way using masked image modeling, learning visual representations from publicly available data that include approximately 30M images. EVA-CLIP is the language-image fine-tuned version of EVA, following CLIP’s training protocol.

2.3 Interpretability Methods

Motivated by the goal of understanding the decision-making of artificial neural networks, the research area of *interpretability* has attracted significant attention. It covers methods that make the model’s behavior or internal states more comprehensible to humans. A full survey of the techniques and frameworks in this area is outside the scope of this thesis, so this section focuses on the methods used in later chapters: attention-based saliency in vision transformers (Sections 4.1 and 6.1.1) and concept-based explanations (Chapter 5).

2.3.1 Attention in Vision Transformers

Attention in vision transformers refers to the self-attention module that is characteristic of this type of architecture [12]. An input image $\mathbf{X} \in \mathbb{R}^{H \times W \times 3}$ is split into $N_{\text{tkn}} = H_p W_p = \frac{HW}{p^2}$ patches, each projected into a D -dimensional embedding:

$$\mathbf{Z}_0 = [\mathbf{X}_{\text{cls}}; \mathbf{X}_p^1 \mathbf{E}; \mathbf{X}_p^2 \mathbf{E}; \dots; \mathbf{X}_p^{N_{\text{tkn}}} \mathbf{E}] + \mathbf{E}_{\text{pos}}, \quad \mathbf{E} \in \mathbb{R}^{3P^2 \times D} \quad (2.11)$$

At each layer ℓ and head h :

$$\mathbf{Q}^{(\ell, h)} = \mathbf{Z}_{\ell-1} \mathbf{W}_Q^{(\ell, h)}, \quad \mathbf{K}^{(\ell, h)} = \mathbf{Z}_{\ell-1} \mathbf{W}_K^{(\ell, h)}, \quad \mathbf{V}^{(\ell, h)} = \mathbf{Z}_{\ell-1} \mathbf{W}_V^{(\ell, h)} \quad (2.12)$$

where $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{D \times d_k}$ and $d_k = D/N_{\text{heads}}$.

$$\mathbf{A}^{(\ell, h)} = \text{softmax}\left(\frac{\mathbf{Q}^{(\ell, h)} \mathbf{K}^{(\ell, h)\top}}{\sqrt{d_k}}\right) \in \mathbb{R}^{(N_{\text{tkn}}+1) \times (N_{\text{tkn}}+1)} \quad (2.13)$$

The spatial attention from the [CLS] token at layer ℓ , head h :

$$\mathbf{a}_{\text{cls}}^{(\ell, h)} = \mathbf{A}^{(\ell, h)}[0, 1 :] \in \mathbb{R}^{N_{\text{tkn}}} \quad (2.14)$$

Reshaped to a 2D map: $\hat{\mathbf{a}}_{\text{cls}}^{(\ell, h)} \in \mathbb{R}^{H_p \times W_p}$.

$$\bar{\mathbf{A}}^{(\ell)} = \frac{1}{N_{\text{heads}}} \sum_{h=1}^{N_{\text{heads}}} \mathbf{A}^{(\ell, h)} \quad (2.15)$$

Attention Rollout

A naive visualization takes the raw attention map $\bar{\mathbf{A}}^{(\ell)}$ of a single layer as the saliency signal. This ignores that information may be mixed across all layers, not just the last one. In addition, the residual connection around each attention block lets tokens bypass attention entirely, so a token's contribution to the class representation is not fully captured by attention weights alone. Abnar *et al.* [46] propose attention rollout to address both issues. The residual stream is modeled as an identity-matrix mixing operation applied to the row-normalized attention, leading to

$$\tilde{\mathbf{A}}^{(\ell)} = \frac{1}{2} \mathbf{I} + \frac{1}{2} \bar{\mathbf{A}}^{(\ell)}, \quad (2.16)$$

where the factor $\frac{1}{2}$ re-normalizes rows. The information flow from the input patches up to layer ℓ is then obtained by composing these matrices along the depth of the network,

$$\mathbf{R}^{(\ell)} = \tilde{\mathbf{A}}^{(\ell)} \cdot \tilde{\mathbf{A}}^{(\ell-1)} \dots \tilde{\mathbf{A}}^{(1)}. \quad (2.17)$$

Taking the row corresponding to the CLS token at the final layer and discarding its self-entry yields the spatial attention map

$$\mathbf{r}_{\text{cls}} = \mathbf{R}^{(L)}[0, 1:] \in \mathbb{R}^{N_{\text{tkn}}}, \quad (2.18)$$

which is reshaped to $H_p \times W_p$ and upsampled to the input resolution. Rollout thus provides a principled, class-agnostic estimate of which input patches most influence the CLS token representation. At the same time, this is a weakness, since the same map is returned regardless of the predicted class, which motivates the method described next.

Relevance Propagation

Chefer *et al.* [47] propose to adapt Layer-wise Relevance Propagation (LRP), proposed by [48] and extended by [49], to the transformer architecture and to fuse it with gradient-weighted attention through a rollout-style update. Starting from the class logit y_c , relevance scores $\mathbf{R}^{\text{lrp}}$ are back-propagated through every operation of the network using LRP-style relevance propagation based on Deep Taylor Decomposition. For every layer ℓ and head h , this yields an attention-shaped relevance map $\mathbf{R}_{\text{lrp}}^{(\ell, h)} \in \mathbb{R}^{(N_{\text{tkn}}+1) \times (N_{\text{tkn}}+1)}$ that is already class-specific. The gradient-weighted attention of the same layer then gates it,

$$\bar{\mathbf{C}}^{(\ell)} = \mathbb{E}_h \left[\left(\nabla_{\mathbf{A}^{(\ell, h)}} y_c \odot \mathbf{R}_{\text{lrp}}^{(\ell, h)} \right)^+ \right], \quad (2.19)$$

so that only attention edges that are both relevant (according to LRP) and sensitive (according to the gradient) remain. The resulting maps are composed across layers with the residual-aware rollout,

$$\mathbf{R}^{(\ell)} = \mathbf{R}^{(\ell-1)} + \bar{\mathbf{C}}^{(\ell)} \cdot \mathbf{R}^{(\ell-1)}, \quad \mathbf{R}^{(0)} = \mathbf{I}, \quad (2.20)$$

and the final explanation is read off from the [CLS] row,

$$\mathbf{e}_c = \mathbf{R}^{(L)}[0, 1:] \in \mathbb{R}^{N_{\text{tkn}}}, \quad (2.21)$$

reshaped to $H_p \times W_p$ and upsampled. Compared to plain rollout, the maps are class-discriminative, and compared to gradient-weighted attention, they are substantially less noisy because LRP accounts for the sign and magnitude of contributions through the full transformer block. This method is adopted as the default saliency signal for the ViT-based architecture in [Section 4.1](#) and its evaluation in [Section 6.1.1](#).

2.3.2 Concept Activation Vectors (CAV)

Kim *et al.* [50] introduced Concept Activation Vectors (TCAV) as a concept-based interpretability method. A user specifies a concept a , for example *striped*, by providing a set of positive examples $\mathcal{P}_a$ and a set of random negatives $\mathcal{N}$. Let $f_\ell(\mathbf{X})$ denote the activations of the network at

layer ℓ . A linear classifier is then trained to separate $\{f_\ell(\mathbf{X}) : \mathbf{X} \in \mathcal{P}_a\}$ from $\{f_\ell(\mathbf{X}) : \mathbf{X} \in \mathcal{N}\}$. Its normal vector defines the Concept Activation Vector (CAV)

$$\mathbf{v}_a^{(\ell)} \in \mathbb{R}^{D_\ell}, \quad \|\mathbf{v}_a^{(\ell)}\|_2 = 1. \quad (2.22)$$

The sensitivity of class k to concept a at input $\mathbf{X}$ is the directional derivative of the logit h_k along $\mathbf{v}_a^{(\ell)}$,

$$s_{a,k,\ell}(\mathbf{X}) = \nabla h_k(f_\ell(\mathbf{x})) \cdot \mathbf{v}_a^{(\ell)}. \quad (2.23)$$

Its main limitations are that it relies on user-curated concept sets and assumes linear concept directions.

Concept Banks

A limitation of pixel-level saliency is that its outputs are hard to relate to a human-understandable vocabulary: a heatmap indicates *where* a model looks, but not *what* it sees. Concept-based methods, such as Testing with Concept Activation Vectors (TCAVs), address this by representing the intermediate activations of a network in a basis of semantic concepts (e.g., turquoise, striped). As adapted by Wu *et al.* [51], a concept bank $\mathcal{B} = \{\mathbf{v}_{a_j}\}_{j=1}^J$ is a collection of CAVs $\mathbf{v}_{a_j} \in \mathbb{R}^D$ in the feature space of a frozen backbone $f(\cdot)$, each associated with a semantic concept label a_j . Given an input $\mathbf{X}$ with feature $\mathbf{z} = f(\mathbf{X}) \in \mathbb{R}^D$, its *concept scores* are obtained by the following projection:

$$s_j(\mathbf{x}) = \frac{\langle \mathbf{z}, \mathbf{v}_{a_j} \rangle}{\|\mathbf{v}_{a_j}\|_2}, \quad j = 1, \dots, J, \quad (2.24)$$

yielding a sparse, human-readable descriptor $\mathbf{s}(\mathbf{X}) \in \mathbb{R}^J$ that can be used for model inspection and debugging. The concept vectors themselves can be learned from small sets of positive and negative examples (Section 2.3.2) or derived from vision-language models such as CLIP [43]. Concept banks form the core of the interactive debugging approach in Section 5.1.

2.4 General Conclusions

This chapter briefly presented various object recognition and detection approaches, different strategies for model pretraining, and interpretability methods for vision transformers that are relevant to this work. From the pretraining sections, it can be summarized that representation learning is a powerful tool for improving downstream performance, both in self-supervised and supervised settings. In addition, as datasets have grown in recent years, stronger and more general backbones and representations have become available. For few-shot learning, such pretrained models offer significant potential, since their inherent visual understanding is already well established and most of the model weights are publicly available. Ongoing

research activity regularly yields improved datasets, architectures, and, in turn, more general transferable representations.

3 Related Work

This chapter provides an overview of three research fields most related to this work: visual few-shot learning ([Section 3.1](#)), current benchmarks for few-shot perception ([Section 3.2](#)), and interactive perception methods ([Section 3.3](#)). It ends with a conclusion in [Section 3.4](#) which addresses the research gaps that this thesis is focusing on.

3.1 Visual Few-Shot Learning

This section presents the related work in the research area of Few-Shot Learning (FSL) in the visual domain. The general idea of FSL is that, similar to humans, new objects are learned from only a few samples - the number is typically restricted to between 1 and 30 examples per novel category. While only few samples are available for novel categories, the architectures are allowed to use base categories with abundant data for preparation. If zero samples are available, the setting is called zero-shot learning and is not within the scope of this work. Zero-shot architectures often rely on other information domains such as text to create category representations. The opposite case, where the number of samples is not restricted at all, is called fully supervised learning. Even there, the amount of training data is not infinite, since it is naturally limited by the effort required to create and handle large datasets. The abundant base categories can be used to either acquire suitable knowledge and representations, or they can be turned into many synthesized few-shot perception tasks so that the model can practice under test conditions. The first approach belongs to transfer learning and the second to meta-learning, which are two of the main strategies.

In general, the difficulty of restricted training data comes from the sparsely sampled distributions of the task-relevant categories. Even in fully supervised learning, dataset variability can be decisive for learning good generalization skills. This variability is heavily restricted in FSL, and compensation strategies are required to still learn the category-specific features and generalize well.

In the following, the few-shot learning challenge is first defined in [Section 3.1.1](#) before current approaches related to this work are covered. Here, a distinction is made between object recognition ([Section 3.1.2](#)), object detection ([Section 3.1.3](#)), and incremental methods ([Section 3.1.4](#)). These research fields are relevant for [Chapter 4](#). The abilities regarding the following questions are assessed:

- How is high intra-class variability handled?
- How do they address fine-grained scenarios with high inter-class similarity?

Finally, a conclusion is drawn in [Section 3.1.5](#).

3.1.1 Definitions and Terminology

In this work’s terminology, few-shot learning refers to the general scenario in the vision domain where models have to adapt to a task based on a restricted number of data samples. The field of Few-Shot Image Classification (FSIC) focuses on tasks where one image contains one main object, usually covering a large portion of the image and with little clutter in the background, and each image is assigned to exactly one category - the one of the main object. Here, clutter refers to objects that do not belong to any of the task-relevant categories. Few-Shot Object Detection (FSOD) on the other hand focuses on tasks where images contain one or several objects, usually with additional clutter in the background. Each object is defined by its bounding box, i.e., the smallest enclosing rectangle aligned with the image frame. In FSOD, an image can contain one or several bounding boxes from one or several categories.

This work focuses on object-centric approaches. Therefore, the term few-shot object recognition aligns better with the research goals than few-shot image classification. Within this work, Few-Shot Object Recognition (FSOR) is considered a subdomain of few-shot image classification, with the distinctive characteristic that image labels always refer to a single, specific object. As a result, labels do not refer to several objects at the same time, nor do they describe an entire scene or a group of objects. FSOD is object-centric by default. Further, the terms *class* and *category* are used interchangeably throughout this work.

Visual few-shot learning can be formalized as follows. There are two sets of categories $\mathcal{C}_{\text{base}}$ and $\mathcal{C}_{\text{novel}}$. The requirement $\mathcal{C}_{\text{base}} \cap \mathcal{C}_{\text{novel}} = \emptyset$ must be fulfilled. For $\mathcal{C}_{\text{base}}$, abundant training data is available. This data is called the base set $\mathcal{D}_{\text{base}}$. The collection of all annotated images for $\mathcal{C}_{\text{novel}}$ is $\mathcal{D}_{\text{novel}}$, and the subset used for task preparation is the support set $\mathcal{D}_{\text{support}}$. The task of few-shot classification is typically formulated under an N -way K -shot setting, where the model is provided with K support images for each of N novel categories that belong to $\mathcal{C}_{\text{novel}}$. For a class $c \in \mathcal{C}_{\text{novel}}$, the number of images $|\mathcal{D}_{\text{support},c}|$ that can be used for task preparation, denoted as K , is typically at most 30. For testing, the model needs to predict an image label y for the query images in the query set $\mathcal{Q}$, referring to exactly one of the N classes.

Similarly, the task of FSOD defines the problem of learning to *detect* a set of N novel object categories $\mathcal{C}_{\text{novel}}$, previously unknown to the model, based on K support images. FSOD also has base classes $\mathcal{C}_{\text{base}}$, which can be used for preparation without any restriction on the number of images per class. For testing, the model needs to predict the object label y as well as the bounding box b , referring to exactly one of the N classes for objects that belong to one of the known categories. In the context of few-shot learning, the term *generalized* refers to the joint consideration of base and novel classes during evaluation. It was introduced by Chao *et al.* [52] for zero-shot learning scenarios, where the test data can contain base and novel classes. In the same way, generalized few-shot object detection takes the performance on the base classes into account after preparing the model for the novel classes. This requires the model to retain

knowledge of the base classes while still being able to learn novel classes [53]. The challenge here comes from the imbalance of data the model has seen for each class group.

Beyond the question of how many samples are available per class, the structure of the categories themselves shapes the difficulty of the task. Two properties are used throughout this work to describe this structure. *Intra-class variability* describes how much the samples within a single category differ from each other. Instances of the same class can vary in pose, scale, illumination, sub-type, surface appearance, or degree of occlusion. *Inter-class similarity* describes how close different categories are to each other in visual appearance. Two classes are similar when their members share a common shape, texture, or part composition, even though they belong to different semantic categories. Both notions are inherently soft and hard to capture with a single objective measure. In this work, they follow the category boundaries that are typically defined by humans and reflect how closely two classes are perceived to be related.

One such case is the *fine-grained* setting, where classes may belong to a common super class and share a certain part of their visual features, leading to high inter-class similarity. Examples include recognizing different bird species [26] or different aircraft models [54]. For a model trained on generic recognition tasks, two birds of different species may look very similar in the internal model representations. The opposite end of the spectrum, low inter-class similarity combined with low intra-class variability, corresponds to coarse-grained recognition with well-separated classes.

After clarifying the terminology, the few-shot learning challenges are now described in mathematical form. Let X and Y denote random variables over images and labels with realizations $\mathbf{X} \in \mathcal{X}$ and labels $y \in \mathcal{Y}$. As mentioned before, in the common few-shot setting, a task is defined by a support set

$$\mathcal{D}_{\text{support}} = \{(\mathbf{X}_i, y_i)\}_{i=1}^{N \cdot K}, \quad (3.1)$$

containing only K labeled samples for each of the N novel categories, with typically $K \in \{1, 3, 5, 10, 30\}$. A model $f_\theta : \mathcal{X} \rightarrow \Delta^{|\mathcal{Y}|}$, often decomposed as $f = h \circ g$ with backbone g and recognition head h , is adapted from $\mathcal{D}_{\text{support}}$ and then evaluated on a query set containing samples of the same categories. The goal is to approximate the conditional distribution $P_{\text{real}}(Y | X)$, although the information available to specify this distribution is much smaller. Looking at the standard fully supervised risk minimization [55] setting

$$\min_{\theta} \mathbb{E}_{(X,Y) \sim P(X,Y)} [\ell(f_\theta(X), Y)], \quad (3.2)$$

which can be approximated by using a finite dataset and the empirical risk $\hat{\mathcal{R}}(\theta) = \frac{1}{n} \sum_{i=1}^n \ell(f_\theta(\mathbf{X}_i), y_i)$. This relies on the assumption

$$(\mathbf{X}_i, y_i) \stackrel{\text{iid}}{\sim} P_{\text{real}}(X, Y). \quad (3.3)$$

and relies on the convergence of θ towards θ^* , which minimizes the real risk $\mathcal{R}$ for $n \rightarrow \infty$. In standard large-scale training, violations of Eq. (3.3) already arise from correlations between samples, e.g., camera types or viewpoints, and from dataset biases such as foreground-background correlations, e.g., polar bear $\leftrightarrow$ snow, leading to $P_{\text{train}}(X,Y) \neq P_{\text{test}}(X,Y)$ and a distribution shift with respect to $P_{\text{real}}(X,Y)$. In the few-shot regime, the problem becomes even more severe: with only a few support samples per class, the empirical distribution based on the support set $\hat{P}_{\text{support}}(X,Y)$ is a poor estimate of $P_{\text{real}}(X,Y)$. In addition, any peculiar property that is frequent within the support set of class c but is not specific to the class, such as a recurring object color or a shared lighting condition, can be picked up as a defining feature of the category. This effect is closely related to the concept of *underspecification*, originally introduced by D’Amour *et al.* [56] to describe training pipelines that return many different predictors with equivalent training-domain performance, but with different behavior during deployment. Building on this, Teney *et al.* [57] use the term *underspecification gap* to refer to the difference between the information represented by the dataset, i.e., the support set $\mathcal{D}_{\text{support}}$, and the information required to perform the task as intended, e.g., according to the requirements defined by an expert. Underspecification allows models with different parameters θ to perform identically on the validation data, while differing on hidden requirements such as Out-of-Distribution (OOD) generalization, e.g., on $P_{\text{real}}(X,Y)$. OOD generalization is one example of a requirement that is not captured by the objective of empirical risk minimization. It cannot be easily measured or enforced, especially when data is limited, but it is a desirable property. For this reason, in-distribution performance metrics alone cannot determine which trained model generalizes best to OOD data [57].

In addition, images are high-dimensional inputs that provide many candidate features for prediction, such as object color, texture, or background. The risk that several distinct features are equally suitable for inference on $\mathcal{D}_{\text{support}}$ is relatively high, even though some of them are spurious rather than causal correlations. Spurious features are defined as features that correlate with a category but are not category-specific [58], and they are picked up by the model for its decision, e.g., a snowy background in images of wolves [59]. They become harmful when their presence without an object of the actual category leads to false positives. Few-shot perception methods are especially at risk. With only K samples per class, the number of features that discriminate the support set typically exceeds the number of features that causally characterize the category. In fine-grained settings this effect is even stronger, since the causal features that distinguish closely related classes are subtle and may be dominated by incidental ones. The following sections present different strategies to deal with these challenges.

3.1.2 Object Recognition Methods

This section presents current object recognition methods that focus on the few-shot setting. Many ideas from the early approaches, such as Prototypical Networks [60], Siamese Networks [61], Model-Agnostic Meta-Learning (MAML) [62], or Matching Networks [63], are still present and are being adapted to more modern techniques and architectures. Seminal work in this field was carried out by Fei-Fei *et al.* [64] in the mid 2000s, who showed that it

is not always necessary to train on hundreds or even thousands of images if the model has been trained on other categories before. Their approach was a Bayesian implementation, and they showed that it had advantages especially when the number of samples was low or even reduced to a single sample per category.

Few-shot object recognition approaches can be divided into the following categories [65]

- metric learning
- meta learning
- data augmentation or generation
- attention mechanism (transformer)
- hybrid variants.

However, a strict categorization is difficult, since architecture modules are combined and methods are mixed. For instance, the popular Matching Networks [63] are commonly assigned to metric learning, but they are trained with an episode-based procedure typical of meta-learning and use an attention mechanism over the support set to weight features. Another typical candidate for metric learning - where the assignment is obvious - are Prototypical Networks proposed by Snell *et al.* [60]. To address overfitting under restricted data, they simplify the inductive bias by using an embedding network and applying nearest-neighbor classification on prototypical class-specific embeddings. In general, this is similar to Matching Networks, where the nearest neighbor method is a weighted variant. The class prototypes are computed as the mean of the support set embeddings for each category. Interestingly, they found that Euclidean distances work much better than cosine distances. Their explanation is that the Euclidean distance is a regular Bregman divergence [66], so the mean class embeddings are the best representative points for classifying or clustering the support set based on proximity. The concept of Prototypical Networks has received active research attention, and many refinements and extensions have been proposed. For instance, Zhou *et al.* [67] address the weakness of prototypical representations in domain transfer scenarios, e.g., *miniImageNet* [63] for training and CUB [26] for testing. They propose an architecture with two branches: one for global, image-level processing and one for local, crop-level processing. While the global branch follows a typical setup, the local branch takes sampled local crops of the query images as input, holds a copy of the global prototypes, and has the task of matching and classifying the local features using them. The prediction probabilities are used to influence the global branch's loss function during meta-training. For testing, the local branch is dropped completely. When pretrained on *miniImageNet* [63], the model's few-shot classification performance on other datasets covering birds, cars, satellite images, and skin lesions, including fine-grained ones, is better than that of other cross-domain approaches.

Instead of optimizing the training of the feature extractor, Shalam *et al.* [68] propose a transformation method to improve existing features for downstream tasks such as classification, matching, and grouping. Their Balanced-Pairwise-Affinities (BPA) feature transform targets the problem that, at test time, the compared feature embeddings have not been part of the

training set, which likely leads to suboptimal representations. With the BPA transformation, they focus less on category-specific information and more on the differences between all features for the task at hand. This makes it especially suitable for fine-grained scenarios. The method starts from the features originally extracted by the backbone for a given set, computes a pair-wise Euclidean distance matrix, and determines the optimal transport (OT) plan while restricting it to the Birkhoff polytope (real non-negative numbers where rows and columns sum to 1). The rows of the resulting matrix are then used as new feature embeddings. BPA can be used as a drop-in module in pretrained networks and classifiers, but it can also be integrated during training, which gives additional, although small, performance gains. The overall approach is transductive, which means that the features for all images in the support set and the query set are processed at the same time. They also propose an inductive variant in which query samples are classified independently and BPA is integrated during training but removed for inference. Their evaluation shows that this variant is strongest in 5-way 5-shot settings, while it is often weaker in 5-way 1-shot tasks.

An early meta-learning approach by Ravi and Larochelle [69] argues that common gradient-based optimizers, such as ADAM [70] or Adagrad [71], are not designed for few-shot scenarios and propose to replace the optimizer with a *meta-learner* Long Short-Term Memory (LSTM). The *meta-learner*, a 2-layer LSTM, acts as the optimizer and predicts the parameter updates for a *learner*, a 5-layer Convolutional Neural Network (CNN). The learner is also the classifier that carries out the actual tasks and receives its task-specific parameter updates from the meta-learner during adaptation. A seminal work in meta-learning is MAML [62]. The main idea is to prepare the model’s parameters through episodic training so that, during the test phase, a short adaptation phase is enough to efficiently adjust them for the task at hand. It outperforms Matching Networks [63] and Siamese Networks [61] on Omniglot [72] and *mini*ImageNet [63].

So far, methods for generic FSIC and fine-grained FSIC have been presented. A separate scenario is addressed by few-shot object instance recognition (FSOIR), in which objects have to be re-identified. For instance, Held *et al.* [73] develop a training setup that optimizes the model’s ability to recognize objects even when only one example image is available. Instead of requiring a 360-degree view of the object, they only need one image. They propose a general-to-specific multi-phase training procedure: first on ImageNet [4] to learn robustness against intra-class variations, and then on the multi-view dataset of object instances called BigBIRD [74]. With the multi-view images, the model learns to produce viewpoint-invariant representations from many or even from a single category sample in the third phase. For final 1-shot training and testing, they use the RGB-D Object Dataset [75].

3.1.3 Object Detection Methods

From a high-level perspective, few-shot object detection approaches can be grouped into methods based on

- detection framework (one-stage vs. two-stage)

- with fine-tuning (gradient-based parameter updates using novel set) vs. without fine-tuning
- base set and novel set handling (e.g., transfer learning vs. meta learning [76, 77])
- major contributions (data-oriented, model-oriented, or algorithm-oriented [78] adapted from [79]).

Few-shot incremental object detection (iFSOD) is similar to FSOD [76], so a short overview of this research area is given in Section 3.1.4. In the incremental learning setting, new classes are added one by one without forgetting previously learned classes.

While FSIC (Section 3.1.2) has been the subject of active research for several decades, Kang *et al.* [80] proposed one of the first architectures specifically for FSOD, which is known as FSRW. They approach the few-shot object detection task using a one-stage detection framework and add a dedicated reweighting module that predicts weight vectors for each category in the support images. In parallel, the feature extractor, a common backbone, processes the input image and provides meta features. Through channel-wise multiplication of the meta features with the weight vectors from the reweighting module, new category-specific features are obtained, which are used for inference and form the input of a final prediction head. The prediction heads estimate one-versus-all objectness and class scores as well as bounding box coordinates. From all class scores, the final prediction is selected by applying a softmax. For evaluation, they propose benchmarks for PASCAL VOC [27] and COCO [28], as well as a dataset transfer setup with pretraining on COCO and testing on PASCAL VOC.

Yan *et al.* [81] propose a two-stage detection model called Meta R-CNN, based on Faster R-CNN [15] and Mask R-CNN [82]. Note that two-stage models have a first stage in which the image is sub-divided into class-agnostic region proposals, which are then processed by a second stage for classification and box regression, as well as for separate mask prediction in the case of Mask R-CNN. Their approach re-uses the Faster R-CNN architecture and adds a parallel Predictor-Head Remodeling Network (PRN), which extracts class-attentive vectors that are used to modulate the RoI-aligned features in the original Faster R-CNN model via channel-wise multiplication. The PRN re-uses the existing backbone and only replaces its first layer with a convolution that accepts 4-channel inputs (image-mask pairs). The architecture is trained with a meta-learning episodic procedure, starting with base classes and continuing with tasks that include base and novel classes. Qiao *et al.* [83] also experimented with Faster R-CNN [15] and analyzed it to improve on its weaknesses for data-scarce scenarios. Their final result is an adapted architecture called Decoupled Faster R-CNN (DeFRCN), which includes two new modules: the Gradient Decoupled Layer (GDL) and the Prototypical Calibration Block (PCB). While components such as the generalized backbone, a class-agnostic RPN, and the classification and regression heads are kept, the new GDL module makes it possible to reduce the impact of, or fully decouple, the backbone from the gradient flow of the other two components during training. The PCB introduces a prototype comparison network, similar to Prototypical Networks [60], which assists in refining the Faster R-CNN classification module. It requires its own backbone pretrained on ImageNet and remains frozen during training and

inference. A method-oriented Two-stage Fine-tuning Approach (TFA) which also uses Faster R-CNN, but without architectural changes, is proposed by Wang *et al.* [84]. In the first stage the model is trained fully-supervised on the base classes. The second stage freezes all weights except for the ones of the box classifier and the box regressor, which are fine-tuned on a support dataset consisting of the few samples of the novel classes extended by an equal number of images from base classes. The fine-tuning on the balanced support set is carried out with a reduced learning rate. For RoI classification they experiment with cosine similarity as the metric and observe that it is often better in low shot settings with 1-5 samples per class than the fully-connected classification variant. Nevertheless, both methods outperformed more sophisticated architectures such as Meta R-CNN [81] and FSRW [80].

The previous examples show that many FSOD methods depend on a fine-tuning stage, even when this stage is restricted to only a small part of the detector head. This can make the preparation for novel classes more complex. For this reason, another group of recent approaches tries to avoid fine-tuning on novel classes [80, 81, 85, 86, 87]. One of the most competitive fine-tuning-free methods is Detect Everything Vision Transformer (DE-ViT), proposed by Zhang *et al.* [88]. It uses a frozen, pretrained Vision Transformer (ViT) backbone to create class prototypes from the support examples and avoids gradient-based adaptation to the novel categories. The authors show that this design choice comes with challenges for existing detector architectures: direct bounding-box regression on the ViT features performs poorly, while unfreezing the backbone quickly leads to overfitting. For this reason, DE-ViT separates proposal generation, prototype-based recognition, and localization refinement. Candidate boxes are first obtained by a class-agnostic proposal mechanism and are then evaluated using similarities between the ViT features and the class prototype vectors. For localization, the method replaces direct box regression with a region-propagation mechanism, where features inside an expanded proposal are used to predict a propagated object region, which is then converted into an adjusted bounding box by a learnable spatial integral layer. This architecture is also used as a foundation for the methods presented in Sections 4.4 and 5.2, respectively.

3.1.4 Incremental Few-Shot Object Detection

The task of incremental Few-Shot Object Detection (iFSOD) was introduced by [89] together with their proposed ONCE architecture which is based on CenterNet [22] - an anchor-based one-stage detection framework. The difficulty of iFSOD comes from the constraint that base classes and their data may not be revisited after the initial model bootstrapping. In addition, novel classes are added one by one using a few annotated samples for each class. The performance is then evaluated on both base and novel classes. The general idea of ONCE is similar to Meta R-CNN [81], with a parallel network that creates class-specific vectors which transform the extracted features in the original architecture (here CenterNet). However, the prediction head, the so-called Object Locator, is class-specific, which allows new classes to be enrolled without fine-tuning. Their evaluation shows that excluding base data during novel class enrollment is challenging and is the reason why many FSOD approaches cannot be transferred to this incremental setting.

Nguyen and Todorovic propose a two-stage iFSOD model called iFS-RCNN [90] based on Mask Region-based Convolutional Neural Network (R-CNN) [82] which is also able to predict instance masks. By introducing Bayesian learning to the classifier and uncertainty-aware box refinement, they achieve good few-shot adaptation skills by preventing overfitting and improving low-shot generalization. Their evaluation shows the architecture’s strength in maintaining strong base class performance, which appears to be invariant to the number of enrolled novel categories.

A method for incremental few-shot instance segmentation that does not require any fine-tuning on the novel classes has been proposed by Zhang *et al.* [91]. The architecture extends iMTFA [92] for incremental Few-Shot Instance Segmentation (iFSIS) with several components that boost the performance on common iFSOD and iFSIS benchmarks - almost catching up with iFS-RCNN, described above, which uses fine-tuning on the novel classes. They argue against using cosine similarity in the classification head as it is done in iMTFA, since it can lead to falsely overestimated similarities, and use a slightly modified version for novel classes.

3.1.5 Conclusions

On a higher level, few-shot learning methods can be grouped into three different types: meta-learning, transfer learning, and data generation. In meta-learning the model is prepared episodically with many few-shot tasks using the base classes, so that it learns to learn under data-constrained conditions. Transfer learning instead relies on strong representational capabilities acquired beforehand. The models develop a general sense for object similarity during pretraining, which is then used through feature-based distance metrics, e.g., in Prototypical Networks [60] and their variations. Data-generative methods multiply the scarce data with image-based augmentation or add completely new samples using generative techniques. A different aspect is whether the model requires gradient-based adaptation on the novel classes. Recent fine-tuning-free approaches such as DE-ViT [88] show that a frozen, large pretrained backbone combined with prototype-based recognition can match or exceed fine-tuned detectors. This demonstrates how prototypes from frozen-backbone representations can be used efficiently.

High intra-class variability is rarely addressed by a dedicated mechanism and is instead handled implicitly. Fine-grained scenarios with high inter-class similarity are more in focus, for instance through the transductive feature reshaping of BPA [68], the local-branch design of Zhou *et al.* [67], or the ProtoPNet family. The evaluations are mostly carried out on bird or aircraft datasets and not on datasets typical for robotic deployment. What only a few few-shot methods offer is a way for the operator to give direct feedback when the support set itself is ambiguous and to resolve the remaining underspecification at adaptation time. The closest candidates to do so mostly belong to the ProtoPNet family, which does not necessarily focus on few-shot scenarios.

Many of the few-shot learning methods that were proposed for FSIC were later also adapted for FSOD, most commonly in two-stage detectors. It is also typical for two-stage models to rely

on a Region Proposal Network (RPN) to obtain bounding box proposals for potential objects. Since the RPN often stays frozen during the preparation with the support set, new objects that the RPN is unfamiliar with might be ignored, and without proposals the recognition part of the architecture is not activated at all. Since items relevant to robotic applications can be rare and uncommon, this poses a potential risk.

Incremental learning methods make the few-shot setting even more difficult. The additional constraint of not having access to any base class data during task adaptation leads to challenges in generalized settings, where the model must not forget base classes while enrolling novel classes. However, in practical scenarios this constraint seems too strict, since a dataset usually only requires a few gigabytes of disk storage, which is not costly and does not require a significant amount of energy. In this work, FSOD methods are considered equally suitable for deployment on robots.

3.2 Few-Shot Perception Benchmarks

This section presents common benchmarks used to compare visual few-shot learning methods. [Section 2.1.3](#) already described typical datasets used for evaluating perception models in fully supervised settings. The following describes and compares the concrete benchmark details. The list of current benchmarks is divided into few-shot classification benchmarks and few-shot detection benchmarks.

Note that common dataset names are often used as a synonym for the corresponding most typical benchmark. Benchmarks often do not have names of their own. However, there are differences. A dataset refers to the annotated collection of data, while a benchmark refers to a standardized evaluation setup. An evaluation setup contains more than just the dataset, and sometimes it combines several datasets. Besides the task definition and the metrics, predefined splits into train, val, and test sets are needed to enable reproducibility and comparison. In few-shot learning, it is often necessary to define the support sets of the test tasks precisely, including the image identifiers, because the per-task performance strongly depends on which images are used in the support set and in the test set.

3.2.1 Image Classification

Omniglot [72] is one of the earliest datasets used for evaluating FSIC models. Described as the *transpose* of the popular MNIST, it contains 1600 characters from 50 alphabets with 20 examples each, instead of 10 digits with thousands of examples each. Today it is less popular, because the resolution is very low at 28×28 and the accuracy of modern models has come close to 100%, which makes a comparison meaningless.

miniImageNet was proposed by Vinyals *et al.* [63] for evaluating their one-shot classification model, because no standardized low-shot benchmark based on ImageNet [4] was available at the time. It contains 60k color images selected from the more than 1.2M images of ImageNet,

covering 100 of the 1000 categories with 600 images per category, which makes the size much more manageable on consumer hardware. However, the image splits were never released, so Ravi and Larochelle [69] released their own, which have since been adopted by many others for evaluating and comparing FSIC approaches. The disjoint train, val, and test sets contain 64, 16, and 20 categories, respectively. While it is more complex than CIFAR-10 [10], the resolution is still rather low at 84×84 pixels. The training categories contain various dog species and several types of birds and insects, which makes the dataset partially fine-grained. With *cliff* or *tobacco shop*, it also includes non-object-centric categories. The popular evaluation protocol of Ravi and Larochelle [69] considers 5-way tasks with 1 or 5 shots and 15 query images per category.

With *tieredImageNet*, Ren *et al.* [93] address some weaknesses of *miniImageNet*. With 608 categories, the variability is increased. Inspired by Omniglot, they group the ImageNet categories into super-categories. Each super-category contains around 10 to 30 categories that belong to the corresponding higher-level type. Instead of splitting the 608 categories into train, val, and test directly, they split the list of super-categories. This ensures that categories of the same super-category do not appear in more than one split. Interestingly, in their evaluation, performance on *tieredImageNet* was consistently better than on *miniImageNet*.

CIFAR-FS was proposed by Bertinetto *et al.* [94], based on CIFAR-100 [10] (Section 2.1.3) but adapted for few-shot tasks. The images have a smaller resolution of 32×32 compared to *miniImageNet*. All categories across the train, val, and test splits can be considered object-centric. The released splits contain fixed categories. However, in [95] a random procedure is assumed for distributing the 100 categories across the splits.

Mangla *et al.* [95], and later others as well [96, 97], included CUB-200-2011 [26] in their FSIC evaluation under a 5-way 1-shot and 5-shot setting, in addition to CIFAR-FS, *miniImageNet*, and *tieredImageNet*. As an evaluation protocol, they use 15 query images per category and run 10K trials in total. CUB-200-2011 is a popular fine-grained dataset that contains 200 bird species (Section 2.1.3).

3.2.2 Object Detection

Bansal *et al.* [98] propose a new split for the COCO dataset [28] that better suits their challenge of zero-shot object detection. They split the classes into 48 seen (train) and 17 unseen (test) classes. The split is obtained by forming K clusters ($K = 10$ for COCO) based on the cosine similarity of the word vector embeddings of all classes. In each cluster, they assign 80% of the classes to seen and 20% to unseen. The images are randomly sampled from the 2014 training set. Because they treat the bounding boxes of the seen classes as foreground and everything else as background, they remove all training images that contain an object of the unseen classes. The whole procedure yields 73,774 training and 6,608 test images for COCO. Although originally designed for zero-shot object detection, this split is also relevant for FSOD benchmarks because it provides a controlled base/novel separation on COCO. The goal of the proposed split was to reduce leakage between seen and unseen categories, which enables a cleaner evaluation of transfer to novel classes under limited supervision.

For the COCO-2017 version of this dataset, Zareian *et al.* [99] propose to use the same class splits, which they call the 48 base classes and the 17 target classes. They then remove any bounding box in the training set that does not belong to the base classes, and afterwards remove the images that are left with empty annotations. This results in 107,761 training images (665,387 instances of base classes) and 4,836 test images (28,538 instances of base classes and 4,614 instances of target classes).

For few-shot object detection, one of the first commonly adopted benchmarks was proposed by Kang *et al.* [80]. They use the PASCAL VOC 2007 test set for testing, describing this as *common practice* for default object detection architectures at the time. The PASCAL VOC 07 and 12 train/val sets provide the training data. They propose to randomly split the 20 object classes into 5 novel and 15 base classes, and keep 3 of these class-split versions. Furthermore, for their COCO benchmark, they use 5,000 images from the COCO 2014 validation set for evaluation (test) and the remaining images in the train and validation sets for training. The 20 classes that overlap with PASCAL VOC are the novel classes, while the remaining 60 are the base classes. They also consider a cross-dataset setting, where the 60 COCO base classes are used for training and the 20 PASCAL VOC classes for evaluation. Their test images may contain distracting base classes that are not targeted in these tasks. They also note that some images do not contain any object of the targeted novel class, which increases the difficulty.

Wang *et al.* [84] also include a few-shot benchmark for PASCAL VOC 2007 and 2012 [27], based on the previous proposal by Kang *et al.* [80], who used fixed sets for PASCAL VOC and COCO. To increase statistical significance, they add more sampled versions to each scenario. They use the train and validation sets of 2007 and 2012 as the training set, and the 2007 test set for evaluation. The 20 classes are randomly split into 15 base and 5 novel classes. In total, they generate 3 random but fixed splits.

3.2.3 Conclusions

Few-shot perception benchmarks often build on popular datasets from the fully-supervised domain. While FSIC addresses instance-level, fine-grained, or generic recognition tasks, FSOD usually targets generic or instance-level detection. CUB-200-2011 [26] is clearly a fine-grained dataset, while others such as ImageNet-1k [4] and *miniImageNet* [63] contain categories with varying granularity, i.e., generic categories such as *street sign* together with several fine-grained dog species. In both cases, the categories typically correspond to common everyday items or to animal and plant species. It should be noted that in some datasets, such as ImageNet-1k and *miniImageNet*, several categories are scene-level rather than object-centric. In this work, recognizing scenes such as *cliff* or *book shop* is not relevant when assessing model performance. While it is part of FSIC, it is not part of few-shot object recognition (FSOR).

Recently, cross-dataset and cross-domain benchmarks [67, 100] have become more popular. The conditions are among the hardest: little data and out-of-domain test conditions. While this is very challenging, it reflects realistic scenarios where the target objects are not represented

in any data that the model has seen before, and adaptation still has to be performed from very few samples.

3.3 Interactive Learning and Perception

This section looks at a particular niche in vision research: machine-learning-based methods that interactively incorporate the human closely into the preparation or operation phase of the perception model¹. Common fully-supervised and few-shot methods also rely on the human. For instance, the collection and annotation of datasets used for training transfers human knowledge about categories or vision tasks into the captured images and labels. Besides that, setting up the training procedure, which includes choosing the architectural components and tuning hyperparameters, can also be a tedious process where the expertise of the user is essential. However, relevant for this work are methods that operate well within a low-shot regime and where the model and human *communicate* and *react* to each other during the preparation or operation phase. The information exchanged and the corresponding interfaces are the focus of the following literature review. In addition, it is examined whether this interaction helps to reduce task underspecification.

Similar methods exist that address scenarios with larger datasets [50, 51, 58]. They typically target spurious features and dataset biases to create more robust classifiers with better in-the-wild performance. For instance, Neuhaus *et al.* [58] propose a technique to reveal and mitigate spurious features for ImageNet [4] classifiers. They incorporate minimal human supervision to detect such features through a PCA-based analysis of the penultimate layer activations. For each class, the most relevant separated features are presented to the operator together with feature-specific visualizations, such as an optimization-based image of the feature, dataset samples with the highest activations, and a GradCAM heatmap for this feature. The operator can then select those features that are not aligned with the target category. To suppress their use during inference, the features are restricted to a non-positive attribution to the class logit. In Chapter 5 some of these methods are transferred to few-shot scenarios.

While the term *active learning* [103] sounds closely related to interactive learning, it differs in important ways. The distinction is as follows. Active learning refers to a procedure where the model's metrics decide which new images get annotated to extend an existing dataset. The overall goal is to annotate as little data as possible while still reaching the required performance. The main differences are that active learning typically assumes abundant unlabeled data for all categories and that the training procedure is repeated multiple times.

¹ The terms *interactive learning* and *interactive perception* are used in other contexts as well, e.g., the first for educational learning concepts that address student participation [101] or the latter for physically interacting with objects or the environment to improve perception quality [102]. Both are outside the scope of this work.

3.3.1 Interactive Methods

Wah *et al.* [104] propose an interactive recognition system that prompts the user with binary questions to help with fine-grained categorization of images. The prompts either ask the user to click on a specific object part or ask about visual part attributes. The approach does not address few-shot recognition, but rather how the human can assist during an interactive inference procedure. The advantages of interactivity are assessed based on recognition speed and accuracy. Chen *et al.* [105] put the model in a more active and independent role for the same task, i.e., CUB-200-2011 [26] classification. Their ProtoPNet model learns prototypical parts for each category while only using image labels. During inference, the parts detected in the query image are compared to all known parts across all categories. The final decision of the model can then be visually explained by showing the detected parts and the most similar samples of the training data. While this improves interpretability, it does not provide a way to intervene in case of false associations. Bontempelli *et al.* [106] address this gap by involving human feedback. Based on ProtoPNet, their ProtoPDebug architecture provides a user interface in which the recognized and decisive parts are visualized as small boxes on the image, and the user can label them via clicks as *use* or *don't use*. This results in two sets of valid and *forbidden* feature concepts, which are used in a subsequent fine-tuning step of the already trained model on a small subset of the training data. During this fine-tuning, part prototypes that are similar to any of the forbidden concepts of a category are penalized, and the model is forced to switch to other prototypical parts. In their evaluation, even non-experts were capable of improving model robustness using this pipeline. However, iterative fine-tuning steps are required to improve the model. Xue *et al.* [107] transferred the conceptual idea of ProtoPNets to vision transformer backbones pretrained on ImageNet-1k [4]. Using the ViT attention rollout (Chapter 2) to obtain a class-specific foreground mask, the model restricts the area that can be used for prototypical parts. While the method presented in Section 4.3 is related, Xue *et al.* [107] do not provide any optional feedback interfaces.

Dehghan *et al.* [108] propose a three-step human-robot interaction pipeline that incrementally learns to detect new objects for downstream manipulation tasks. On a touch-based screen, the operator can activate a teaching mode in which new objects are simply held into the camera view to automatically obtain support images. The label for each new class must also be provided. For their incremental FSOD they incorporate the RPN from the Faster RCNN [15] architecture pretrained on COCO. As the RPN backbone they use VGG-16 [7] pretrained on COCO [28]. In the teaching phase, they rely on saliency-based class activation maps (CAM) [109] from ResNet-50 [9] (pretrained on ImageNet [4]) to localize the relevant image regions and extract their features. The mean over these regions across all presented object poses serves as the identifier for the new class. During inference, the classification is a simple k-NN [110] retrieval of all known classes for each proposed bounding box from the RPN. They compute the ratio between the nearest and second-nearest category distance. If it is clearly below a threshold, the object is assigned to the nearest category, otherwise it is treated as an unknown object. In that case, the user has no way to fix this except by repeating the teaching

phase and hoping the features will be more distinct. Berg *et al.* [111] propose a similar framework for image analysis applications in bioinformatics. However, instead of using modern few-shot approaches, they restrict the model complexity, for example by using Random Forests [112], with the benefit of supporting non-expert users and standard consumer hardware.

Bohus *et al.* [113] proposed an incremental and continual learning system that uses data and labels acquired through a mixed-reality headset. They use a multi-component system with an integrated object tracker and a speech recognition module to enable voice commands. These commands can, for instance, be used to introduce a new object to the system. Using class-agnostic 3D object detection together with the current gaze of the user, the new label is assigned to the object. Subsequently, multiple views of the same object are acquired by going backward and forward in time. The system checks the records for previous views of the object from other angles, and for future views the object location is stored and tracked. Whenever the object is computed to be within the user's field of view, an image is saved together with a bounding box label inferred from the projection of the 3D object point cloud into the camera frame. This way, the user neither needs to walk around each object to obtain training data nor has to label it again. They used Contrastive Language-Image Pre-training (CLIP) [43] for zero-shot recognition by querying it with $(image, [c_1, c_2, \dots, c_N])$ tuples and obtaining the probability for all N known categories. In addition, they trained a multinomial logistic regression model on the encodings of CLIP's visual encoder to support continual learning.

3.3.2 Conclusions

Various types of interactive interfaces were proposed in the presented approaches. These range from simple click-based interfaces over multi-panel touch-screen applications and full-blown research-ready assistive software [111] to modern mixed-reality devices that enable high mobility. While the interfaces and the architectures are both intuitive and effective, the focus is typically on common household items, animals, or plants.

A suitable benchmark for interactive learning methods would cover tasks where the support set is identical across conditions and the query images are the same, so that the effect of user involvement can be measured directly. However, the assignment of images to categories is mixed across conditions. This is possible because the support set, with only few samples per category, is underspecified, i.e., the images in each class-specific support set have several unique but also shared features with at least one other category. No such benchmark was found during the literature review.

3.4 Summary

This chapter presented related work on visual few-shot learning, discussed a diverse range of benchmarks, and introduced existing strategies for incorporating humans into the preparation

and operation phases. Detailed conclusions are provided after each section. In the following, the findings are summarized and the research gaps are outlined.

Few-shot learning is an active field of research, and its motivation remains relevant. Fully-supervised perception models require large datasets. In many scenarios, including typical robotic applications, this is not feasible, and architectures that can learn a novel category from only a few samples are required. Elaborate approaches with strong performance on common benchmarks have been proposed in the literature.

The common categorization into meta-learning approaches and transfer learning is not that strict, since architectures and components of both types are increasingly interchanged. In general, however, transfer learning methods often have simpler setups and strong performance, even with fine-tuning on the support set. These methods leverage pretrained models and their representations for few-shot settings. With the recent growth in publicly available backbones and models pretrained on large-scale datasets, this represents a promising combination. Therefore, this work investigates the capabilities acquired by such models, focusing on realistic object-centric robotic applications within a low-shot learning regime. A recurring pattern across the surveyed detectors is the reduction of each class to prototype vectors that interface a frozen backbone with a lightweight recognition head, which also motivates the multi-granular prototype design of [Section 5.2](#).

Sometimes even humans can only guess the *real* task or target category from a few samples. In such scenarios, context knowledge becomes valuable. Therefore, this work also adapts the idea of extending such architectures with interactive learning, with the goal of incorporating the user’s guidance through suitable interfaces to resolve ambiguities, especially in scenarios with high inter-class similarity.

4 Pretraining-Based Few-Shot Perception

This chapter introduces the developed methods for object-centric recognition under practical constraints, especially limited data, that all build on pretrained backbones. [Chapter 5](#) will then present interactive procedures that can improve model quality with human feedback. [Chapter 6](#) will evaluate both the methods from this chapter and the interactive approaches from [Chapter 5](#).

Optimizing a perception model for one specific task usually requires setting up a corresponding training dataset and then optimizing the model's parameters on this dataset. As described in the introduction ([Chapter 1](#)), this is often not feasible in robotic applications for various reasons. For instance, there is neither enough data nor enough time to provide a sufficiently large training dataset. This chapter presents methods that address this challenge by reusing the capabilities of pretrained models in different ways.

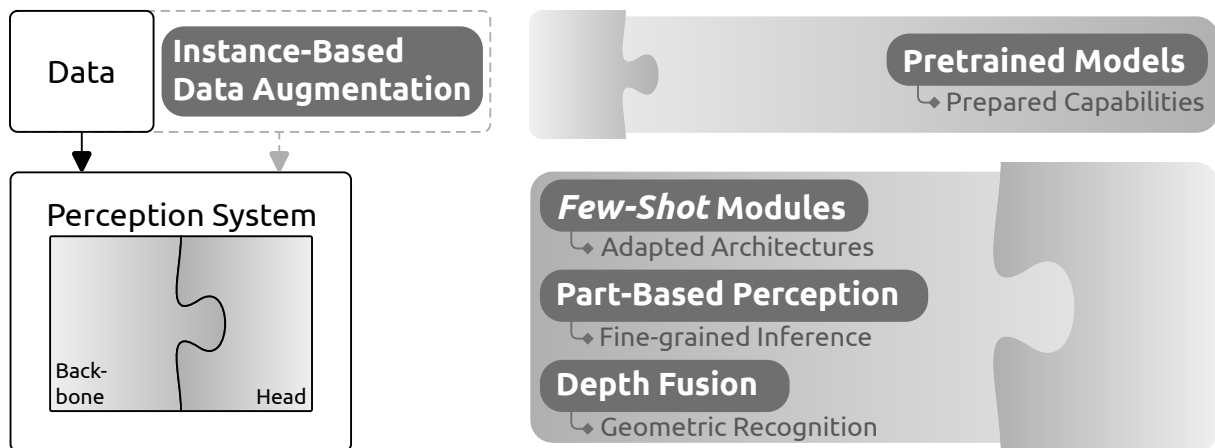


Figure 4.1: The schematic provides a simplified overview of the concept for perception systems covered by this chapter. A typical few-shot learning strategy uses pretrained vision models (commonly split into backbone and head) to reduce data requirements, and then fine-tunes them on a dataset specifically created for the task at hand. This procedure is adapted and extended in various ways, e.g., by adding newly developed modules or by focusing on part-based pipelines for fine-grained recognition. One method incorporates depth information to handle scenarios where geometric recognition is beneficial for robust perception. Complementary to this, instance-level data augmentation allows mission-relevant backgrounds to be incorporated while artificially increasing the number of training samples.

Today, many pretrained models are publicly available online with permissive licenses. They cover a wide variety of architectures, usually trained on the most common datasets for different vision tasks. The underlying idea is that these pretrained models have acquired a set of capabilities that can serve as a starting point, or that can even be applied directly to the new task at hand. This idea is well established and is known as transfer learning. The methods

presented in the following sections combine transfer learning with several other techniques to adapt the models to typical robotic applications. While each development targets a certain application type, the methods can also be used in other applications that are similar to the ones presented in this work. An overview of the chapter's concept is depicted in [Fig. 4.1](#).

4.1 Frozen Vision Transformers for Attention-Based Segmentation

Vision Transformers (ViTs) trained on large-scale classification tasks develop rich internal representations, and their attention mechanisms learn to identify meaningful regions within images, encoding implicit knowledge about what constitutes interesting visual patterns. While ViTs were initially developed for image classification [12], their use for segmentation tasks is also an active research field [114]. However, this usually involves elaborate training procedures. Instead, the core idea of the approach presented in this section is to exploit the understanding of visual relevance embedded in the weights of pretrained models and to repurpose it for other tasks without additional training. Rather than learning explicit models of specific object categories for semantic segmentation tasks, the attention of a pretrained vision transformer serves as an indicator for instances that deviate from the rest of the image. Zhang *et al.* [115] also use attention masks for segmentation, but they add a decoder-like network with a specialized module that converts attention maps into class-specific segmentation maps. All of these components require task-specific training. In contrast, the proposed method is simpler and builds directly on ViT attention maps, removing the need for additional training. This concept is suited for scenarios with categories that show high intra-class variability, where limited data and training cannot capture the full range of characteristic features. The specific scenario used for later evaluation is the segmentation of litter items lying on the ground in outdoor environments. Here, the implicit properties of objectness and saliency are exploited, as these items typically appear visually distinct from the rest of the scene. This enables segmentation of potential trash objects without specific training and provides inherent generalization to unseen items. However, the approach should not be seen as an immediate replacement for other litter segmentation models, but rather as an exploratory study.

Parts of this section were previously published in [P1] and build on a supervised thesis [T8].

4.1.1 Target Scenarios

The practical real-world scenario that motivates this research is the idea of fleets of mobile robots for outdoor litter cleanup, similar to the Angsa Robotics startup¹, which aims to reduce the accumulation of waste in natural environments. While the startup and many others [116] rely on large datasets with precisely defined classes of waste items that are continuously updated and extended, this work takes the inverse approach and defines waste items as salient regions in the image that are not associated with any known non-waste category. The goal is to avoid the need for large litter datasets, which can never be complete given the endless variability of waste items. The litter segmentation application conditions are also used for evaluation in Section 6.1.1. However, the concepts may be transferable if the following criteria are met:

¹ <https://angsa-robotics.com>

- object-centric segmentation tasks
- a common background (e.g., soil, pavement, or grass) similar to those in typical classification datasets
- high intra-class variability and limited available data

4.1.2 Architecture and Method

The proposed architecture uses a pretrained vision transformer [12] as its backbone and exploits its inherent understanding of visual relevance and objectness. Unlike convolutional neural networks, where receptive fields gradually grow from layer to layer, vision transformers process the global image context from the first layer through their self-attention mechanisms. This property allows them to identify relationships between distant image regions at an early stage, which helps to distinguish foreign objects from their environmental context.

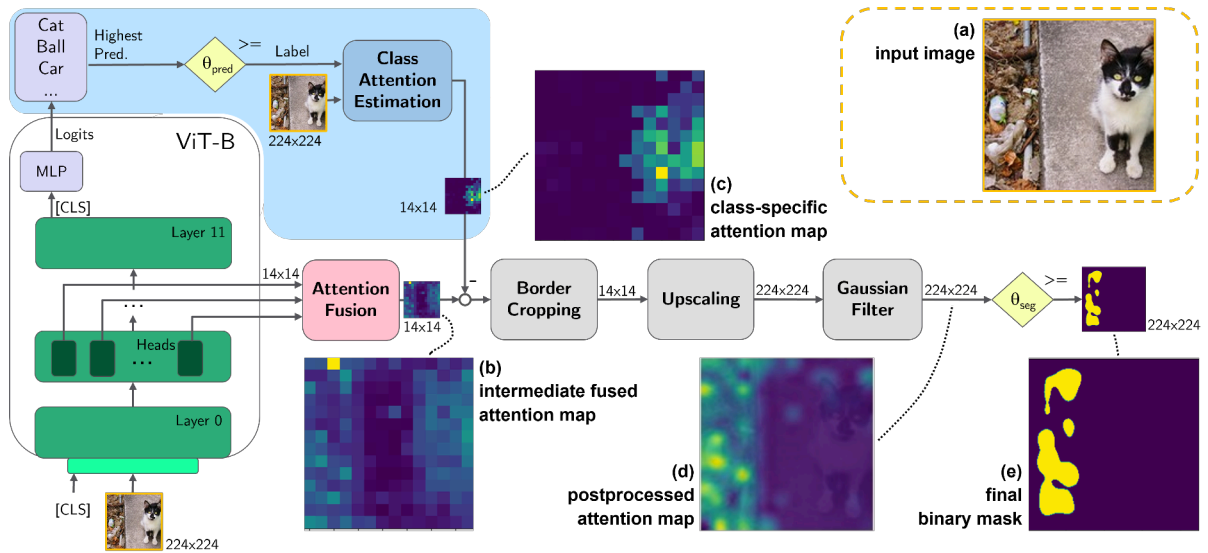


Figure 4.2: Overview of the proposed modular segmentation architecture. The pipeline builds on a pretrained ViT-B backbone [12], where an *Attention Fusion Module* connects to an intermediate transformer layer (layer 4, Section 6.1.1) to extract and merge attention maps across all heads. The optional *distraction branch* (shown in blue) computes class-specific attention using CLS-LRP [47] and subtracts it from the fused attention map to suppress regions that correspond to recognized non-trash objects. Subsequent post-processing stages, including border cropping, upscaling, and Gaussian filtering, refine the spatial resolution and quality of the attention map before threshold-based binarization produces the final segmentation mask. Copyright ©2025, IEEE [P1].

Fig. 4.2 shows the full processing pipeline. The developed architecture takes RGB images at 224×224 resolution as input and produces binary segmentation masks at the same resolution. These masks mark regions that correspond to potential litter objects. The current version of the pretrained vision transformer, which is the core component, is a ViT-B model pretrained on ImageNet-1k [4]. It consists of 12 layers with 12 attention heads per layer. The modular design of the overall architecture allows substitution with alternative vision transformers, so that different computational budgets or resolution requirements can be met without major

adjustments. Note that no task-specific fine-tuning is performed on the transformer weights, so its parameters remain fully frozen.

The vision transformer processes input images by first dividing them into non-overlapping patches of 16×16 pixels, which yields a 14×14 grid of patch tokens. An additional classification token is prepended to the sequence and serves as an aggregated representation that is later used as input for the classification head. The 12 attention heads in each layer operate independently, so each head focuses on different features and spatial relationships. Their attention maps have the same resolution as the patch grid, i.e., 14×14 . The Attention Fusion Module (AFM) aggregates the attention maps across all heads within a selected transformer layer into a single map. The choice of layer plays an important role, since the attention in each layer focuses on different image areas. The evaluation section ([Section 6.1.1](#)) explains why layer 4 is chosen for this work. An optional Distraction Branch computes class-specific attention, which visualizes the regions that lead to the classification decision. In this way, those regions can be suppressed within the fused attention map if the estimated category corresponds to a non-trash object. Finally, a post-processing pipeline based on conventional image processing methods refines the resulting attention map and produces the final segmentation mask.

Attention Fusion Module (AFM)

The Attention Fusion Module (AFM) combines the head-specific attention maps into a single map that represents the overall image saliency for the given layer. The attention patterns vary across layers, and they also differ between the heads within each layer. For this work, different fusion strategies are tested to obtain a final map for further processing: minimum, maximum, and mean aggregation across heads. Future research could investigate the role of each head in more detail.

The module can be connected to any layer within the transformer architecture. As mentioned before, in the experiments layer 4, i.e., the fifth layer, combined with mean fusion showed the best performance. This suggests that intermediate representations capture a suitable level of abstraction for the evaluated litter segmentation task.

Distraction Branch

The Distraction Branch is introduced to address scenarios in which known non-litter objects, such as animals or other visually salient elements, are present in the scene and attract the model's attention. Such situations can occur in real-world settings. For instance, a cat may appear in the image. Since cats are part of the ImageNet-1k dataset, the model usually focuses its attention on them, which causes the segmentation to incorrectly predict them as litter. From a practical perspective, such issues could also be mitigated by adjusting the camera viewpoint or height so that only ground regions are captured. In contrast, the proposed module aims to improve internal robustness during inference. Note that this branch is currently optional and is used mainly for qualitative analysis.

The technical implementation of the Distraction Branch relies on Class Attention Estimation (CAE), which uses different strategies to determine the regions that belong to known categories. One of them is the contribution estimation introduced by Chefer *et al.* [47] (Section 2.3.1), a Layer-wise Relevance Propagation (LRP) method developed specifically for transformer architectures. The technique estimates the contribution of each input patch to the final classification decision through the network’s attention maps. The result is a 2D map that indicates patch-level relevance for the predicted class. The other strategies tested are simpler and are based on the fused attention map itself. The pretraining dataset of the vision transformer contains 1000 categories, but the branch is only activated for those with a prediction score above a certain threshold θ_{pred} . The CAE map is then used to mask regions in the fused attention map obtained directly from the layer. This lowers the saliency of regions assigned to recognized objects and allows the model to redirect its focus to salient regions that contain unknown items.

Post-Processing Pipeline

The post-processing pipeline transforms the raw attention-based saliency map into a binary segmentation mask through a sequence of operations. Each stage addresses limitations of the attention extraction process and gradually improves the mask quality for the segmentation task. Border cropping sets all attention values along the outer image edges to zero, since they tend to contain artifacts in the form of unusually high attention values that do not correspond to actual objects. Empirically, cropping these regions improves segmentation precision without harming the detection of centrally located trash objects. Upscaling then resizes the attention map from the patch resolution back to the original image dimensions. The 14×14 patch grid yields relatively coarse spatial information, so conventional interpolation methods are applied to restore the image resolution. The subsequent Gaussian Filter module addresses the pixelated effects of upscaling by smoothing the upscaled map, rounding edges and producing more natural region boundaries. The final binarization step applies a threshold to convert the continuous map values into a discrete binary segmentation mask.

4.1.3 Model Preparation

The model preparation does not require any additional fine-tuning on litter datasets. However, the proposed modules have several internal parameters. The default values determined during the evaluation on the TACO [117] dataset (Section 6.1.1) can be used directly, but they may need to be adapted to the task at hand, either with an available dataset that matches the planned application environment or with a new one, which does not necessarily need to be large.

4.1.4 Inference and Operation

Since the vision transformer and the integrated modules are not demanding in terms of computational power, inference can be performed even on smaller or embedded Graphics Processing Units (GPUs). Due to the low image resolution, the memory footprint is also manageable.

During operation, the tunable internal model parameters are not meant to be changed, but they can be adjusted if malfunctions are detected. The model predicts binary segmentation masks that may contain multiple litter objects. On real robots, where cleanup actions are object-centric (e.g., gripper-based) rather than region-centric (e.g., vacuum-based), such masks may be a significant limitation.

4.2 Transfer Learning with Pretrained Detection Models

This section presents an approach for low-data object detection that combines multiple conventional methods. The starting point is a common object detection architecture. Through transfer learning and tailored augmentation strategies [118, 119, 120], the amount of labeled data required for training can be significantly reduced. Copy-paste augmentation methods typically focus on tasks involving individual object instances [118, 119].

In contrast, the approach presented here addresses generic category-level detection from only a few samples and is inspired by the work of Ghiasi *et al.* [120]. In their study, the authors artificially restricted the amount of training data in COCO [28] before re-augmenting it for standard fully supervised training. To compensate for the substantially lower number of samples per class used in this work, depth-processing modules are added to improve robustness and to let the system use geometric differences when inferring object categories.

The incorporated methods support the integration of models for object detection and instance segmentation. Thus, this method is suitable for robotic applications where object masks are required as outputs, e.g., for subsequent robotic grasping of the detected objects.

Parts of this section were previously published in [P5] and [P8], and reported in [P7].

4.2.1 Target Scenarios

The presented approach has been validated in two distinct robotic applications. The main target scenario is an automated battery sorting line using an industrial robot. For this scenario the incorporated depth processing was essential. Detecting batteries belongs to a challenging group of tasks with high intra-class variability. The reason is the wide variety of visual designs chosen by the manufacturers. In addition, the inter-class similarity to other categories can be very high since manufacturers reuse the same designs across different battery types. Often the only features that make them distinguishable are either text printed onto the battery, which can be scratched off and unreadable in recycling plants, or their geometric shapes, which remain mostly standardized within a category.

When applying the approach to the second scenario, i.e., the retrieval of hazardous objects using a mobile robot platform, the enhanced augmentation strategies were valuable because the starting situation was different. The intra-class variability of the categories was lower, but the variability of the possible backgrounds was rather high.

The approach is assumed to be transferable to other applications that match the characteristics of the evaluated scenarios:

- limited availability of training data (e.g., in case of rare objects)
- requirement for instance-level segmentation mask outputs
- operation in cluttered environments not fully represented in standard datasets

- high inter-class similarity for categories that have different geometric shapes.

Categories with high intra-class variability make preparing datasets harder because this variability must be reflected in the training set, which in turn leads to larger datasets.

4.2.2 Architecture and Method

The developed architecture uses RGB images and optional 3D point clouds as inputs. Figure 4.3 depicts the system components and the overall functionality. The foundation for the transfer learning is a pretrained Mask R-CNN model, introduced by He *et al.* [82], which processes the images. It is a two-stage detector: first it generates region proposals using a Region Proposal Network (RPN), and then it classifies these proposals and refines their bounding boxes. The mask predictions are produced simultaneously in a parallel network branch and use the aligned Region of Interest (RoI) features to predict instance masks. Since the Mask Region-based Convolutional Neural Network (R-CNN) predicts masks independently of the classification results, the mask prediction is more class-agnostic, which likely helps when fine-tuning on smaller datasets. In theory, the Mask R-CNN could be replaced by any other detection model within the overall system architecture, even one that is not capable of instance segmentation. However, the benefit of precise segmentation is that objects can be arranged much more densely in the images while the rest of the architecture still works as intended. The reason is that, due to the pixel-level masks, the 3D information from the point clouds can be assigned much more precisely to a given detection of the vision model. Having only the common unaligned bounding boxes as model output would make this challenging in the case of occluding objects, or smaller objects that overlap with the bounding boxes of larger objects.

Transfer Learning Methodology

The transfer learning process uses pretrained weights from models trained on larger datasets such as COCO [28], a comprehensive dataset containing more than 300k labeled images across 80 common object categories. Pretraining the vision model on a large general dataset produces weights that encode general visual recognition capabilities, which can then be transferred to new specific tasks.

In this case, the basic object detection and instance segmentation capabilities acquired on COCO are already encoded in the model's parameters. Fine-tuning on a dataset for a new task then aligns the parameters to the task at hand, requiring less data and fewer iterations for convergence. It starts from the capabilities for general objects such as the categories in COCO and adapts them to work with the potentially very specific target objects of the new task.

For fine-tuning, the first layers can be kept frozen, such as the feature extraction backbone, so that parameter optimization only targets the detection and mask heads of the model. This reduces the computational requirements of the fine-tuning procedure by lowering the memory demand and the number of optimization steps. The lower layers responsible for less abstract

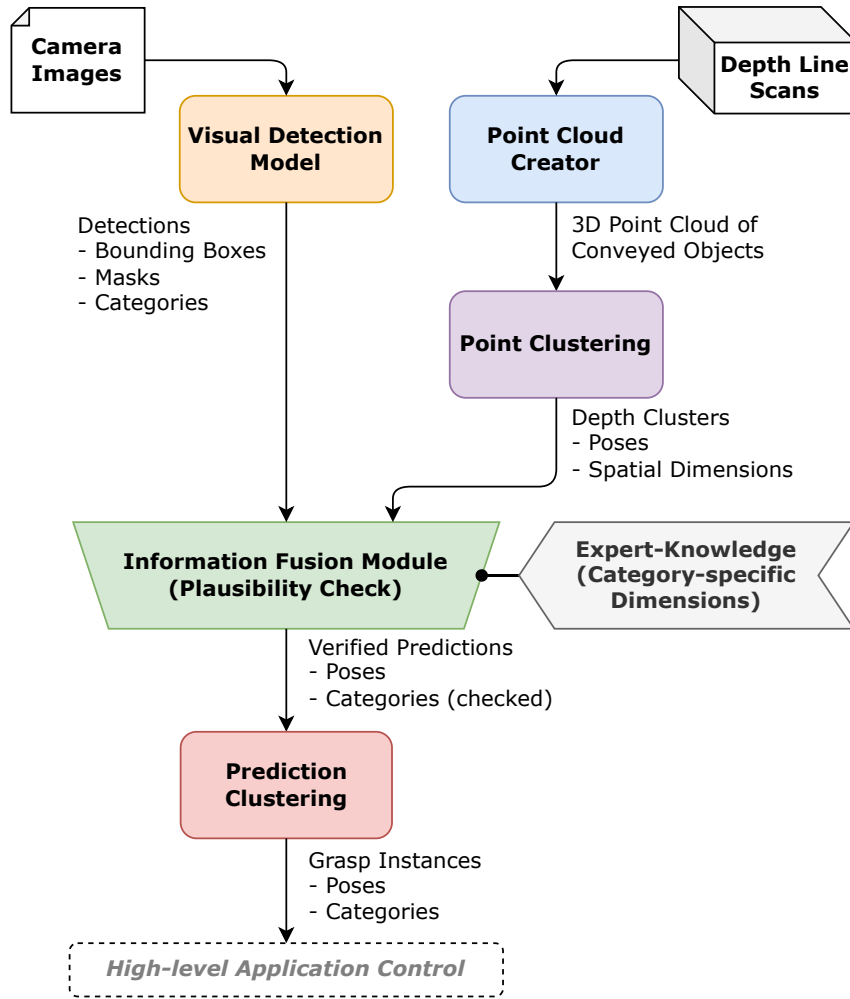


Figure 4.3: Overview of the hybrid prediction architecture. The Information Fusion Module (IFM) (green) takes the geometric rules set up by the user as input. It validates the visual detections according to the rules and the depth data that was perceived simultaneously in a parallel branch. Copyright ©2025, IEEE [P5].

features remain unchanged. Note that training with too few iterations risks insufficient adaptation to the new categories. On the other hand, excessive training leads to overfitting the model parameters to the instances in the provided images. The model can become too specialized on the limited training samples and lose generalization to instances of the same category that have a different visual appearance. Similarly, learning rate scheduling plays a critical role. A reduced learning rate is typically recommended compared to training from scratch, to prevent catastrophic forgetting of the pretrained capabilities.

For the detection heads, the final classification layer is special and cannot be pretrained, because the final output layer depends on the number of categories, which is usually different for a new task. This layer is therefore initialized with new parameters and is essentially the only component that is learned *from scratch*, even in a fine-tuning procedure.

Information Fusion Module (IFM)

The IFM is one of the core components of the developed system for detecting custom objects with only little annotated data. It fuses the initial visual predictions based on camera images with the geometric information obtained by a separate depth processing branch. The goal is to improve the robustness of the visual perception and to combine geometric with visual information. It verifies the predictions based on expert knowledge, which has to be provided in the form of rules for category-specific geometric dimensions.

Due to the lack of data for training the vision model, the categories for the task at hand cannot be specified sufficiently. Therefore, the idea is to adapt the detection model towards the new categories by fine-tuning on the available data, while carefully avoiding overfitting and instead retaining a higher level of generalization that matches the higher variability of the categories underrepresented in the small training dataset. This is achieved by keeping beneficial inductive biases intact, which were developed by the model in the extensive pre-training phase. With a vision model that will detect even unseen objects but at a higher risk of confusion, the system can then correct these false predictions by applying expert rules for verification in the geometric domain.

The IFM takes the list of visual detections and the processed information about the derived point cloud clusters as input. A detection contains the bounding box, the binary mask, and the predicted category, whereas a point cloud cluster is defined by its 3D position and its spatial dimensions. In the current version, a detection is assigned to a cluster by projecting the cluster centers into the camera frame and determining the nearest neighbor to the detection center. Note that the depth processing can be highly application specific and depends on the available depth sensors. What matters is that the output of the depth processing branch provides meaningful geometric data that helps to distinguish the categories of the task at hand. For this, a human expert needs to set up a table of rules that reflects the category-specific geometric features. For instance, the minimum height, the maximum length, or the expected volume. These rules are designed to be easily configurable, even by people who are not machine learning experts but who are highly knowledgeable about the task at hand. For each rule, the operator needs to configure when it should be applied for verification and what happens when the verification fails. An example: the length is checked when the visual prediction is *Class A*, and if the verification fails, the predicted category is changed to *Class B*. If no rules are configured at all, the visual predictions are extended with the geometric information from the assigned depth clusters, but the categories are kept unchanged. This means that the vision model is fully trusted or that no confusions are expected that could be resolved through the derived spatial dimensions. The rules are created empirically by the operator, either before or during operation. Adding rules during operation offers the useful feature that the rule set can be completed online based on the actual performance of the vision model. Note that when one class is often confused by the vision model with another class, but there is no way to tell them apart based on their geometric dimensions, the system must return to the fine-tuning procedure and adjust either the dataset or the hyperparameters.

By design, vision models are typically invariant to scale, because scale usually just reflects a different distance between the object and the camera. However, especially in industrial use cases, many objects or parts are handled separately depending on their geometric size, e.g., batteries in a recycling plant. This is where the hybrid inference using depth information comes in and provides a capable option. Another way to exploit the hybrid inference is to use only one category of visually similar objects, e.g., special metal parts such as female screws, when creating the visual training dataset. Since the intended sub-categories, i.e., different sizes, are separable by their dimensions, they can be split again by creating the corresponding verification rules. This way, the image dataset needs fewer categories and therefore fewer samples as well.

One could argue that using only depth might be enough in some cases and that the vision model is not required at all. That may be true for some applications, but the goal was to develop a system that can be quickly adapted to various applications, and the vision model allows objects with the same shape but different appearances to be distinguished, which offers more flexibility. In addition, partial occlusions are harmful for geometric depth processing, whereas vision-based perception models can still recognize the correct category.

Data Augmentation Strategies

Data augmentation expands the available amount of labeled data by multiplying it through transformations and modifications. For few-shot scenarios, augmentation can help to better avoid overfitting while extracting as much information as possible from the limited data. Standard geometric 2D augmentations include random horizontal flipping, rotation, scaling, and cropping. Photometric augmentations such as brightness, contrast, and color jittering increase robustness to diverse illumination conditions or even different image sensors. While these modifications use the available images, other approaches integrate new images as backgrounds to create new object-background combinations [118, 121, 119, 120]. This is done by cropping instances based on their pixel-accurate mask annotations and then pasting them onto new backgrounds.

In the presented approach, such methods are used to prepare the model for the expected operating environment. While images of the target objects might already exist or can be quickly captured, it is often not feasible to create images in an environment similar to the one of the planned mission, especially in the case of mobile robots. Examples are dangerous or hazardous environments that prohibit humans from entering, or missions that are time-critical and whose environment is only known shortly before operation. For stationary robots, it might also happen that, for example, a conveyor belt has to be exchanged with a new one that has a different appearance or texture, which can lead to falsely recognized objects. Creating new datasets with the appropriate backgrounds can be tedious, even when the datasets are small. Cluttered backgrounds in particular can lead to many false positives. This can be prevented when the model is familiar with the environments and the typical items within them. Therefore, to

extend the small dataset created by the user, default 2D geometric image augmentations are combined with cut-and-paste strategies.

The operator can add mission-specific background images, which must not contain any of the target objects. The backgrounds can be assigned to one of two levels, *default* or *hard*. The second group is sampled more often when synthesizing new images with cropped objects. The operator can further increase robustness by including similar negative objects as part of the backgrounds, since the model also learns from negative examples during the fine-tuning phase.

Note that augmentation strategies do not solve all the disadvantages of small datasets. Excessive augmentation does not change the fact that the intra-class variability of small datasets usually does not reflect the real world. The augmented samples are ultimately derived from the same limited source images, which constrains the diversity of representations the model can learn from the dataset. In applications where 3D meshes are available or obtainable, e.g., through visual 3D reconstruction or depth-based sensors, the randomized texturing proposed by Hönig *et al.* [122] is not only a promising and efficient data augmentation technique, but may also enhance the shape bias of the detection model during fine-tuning, if it is suitable for the task at hand.

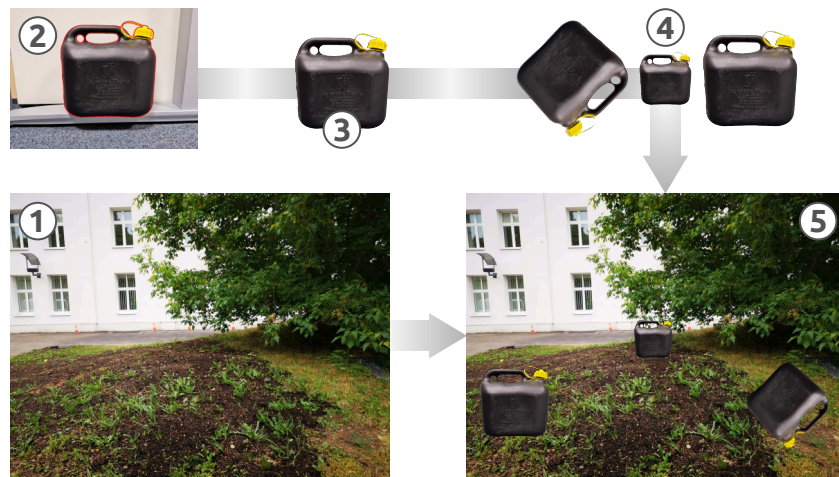


Figure 4.4: Simplified overview for the instance-based dataset augmentation. The augmented image ⑤ is created by labeling a new image ② or by taking an available dataset element. After cropping the target object ③ various augmentation methods are applied, e.g., rotation, size, or brightness, before the modified versions ④ are pasted onto the desired background image ①. During pasting the object edges are merged into the background using Gaussian filtering along the edges of the pasted crops. The new image annotations are automatically generated. Only a subset of the augmentation methods is depicted.

4.2.3 Model Preparation

Preparing the system for deployment requires creating a task-specific dataset and running the fine-tuning procedure. The dataset must contain representative objects with accurate mask annotations for all target categories. In general, the precision of the annotated instance boundaries improves the final quality of the predicted masks. Collecting diverse samples for each

category should be prioritized over quantity, since the applied data augmentation strategies can compensate for low quantity better than for a lack of sample variability. This means that taking different images of the same object with variations in viewpoint, lighting, and background is not as valuable as adding an image of a different item from the same category.

The preparation therefore depends on the type of target objects. For objects resembling COCO [28] categories, the pretrained detection model might be enough and no additional data needs to be created and annotated. In contrast, very specific object types with distinct visual characteristics tend to require more samples to establish reliable feature representations.

The subsequent fine-tuning, which follows the data augmentation step, has lower computational demands than training the complete vision model, since most of the lower layers can be kept frozen. Furthermore, already known rules for the geometric verification within the IFM can be created beforehand, but they can also be added at any time, even online during operation.

4.2.4 Inference and Operation

During inference, the images are processed by the fine-tuned detection model. The inference requirements depend on the selected architecture and the temporal constraints of the robotic application. Without a GPU, the processing time of the Mask R-CNN (Mask R-CNN) model is far from real-time operation. In parallel, the depth processing pipeline continuously provides object-related information suitable for fusion with the visual detections, and it usually runs faster than the perception model, except when the point clouds are large. Note that the fusion pipeline requires a calibration of the camera with respect to the depth sensors and the robot, respectively.

As output, the system provides the visual detections, potentially corrected by the IFM, and extended by the spatial dimensions obtained from depth processing. The operator then has multiple options for adjusting the system's performance during operation to reduce the number of confusions or false positives. The rules already instantiated for validation can be modified or extended with new ones. This can be done online during operation. Naturally, the verification step becomes more complex as more rules are added, but this has a minimal effect on runtime or computational requirements.

If false model predictions occur consistently due to a certain object or part of the background, and they cannot be resolved by geometric rules, the only option is to return to the augmentation phase: either the background is added to the *hard* backgrounds, or the number of images for the two confused categories is increased. This can be done by capturing new images or by increasing the number of augmentations.

4.3 Part Prototypes for Fine-Grained Few-Shot Object Recognition with Frozen Models

This section covers a system that uses an extensively pretrained model as its central core. The previous sections also used pretrained models, but in [Section 4.2](#) they go through a fine-tuning procedure, and in [Section 4.1](#) they do not recognize specific categories but rather general top-level supercategories. In contrast, the system presented in this section can recognize fine-grained object categories and also provides interactive interfaces that the operator can use to adjust the model’s behavior. Note that everything specific to the interactive properties is covered in depth in [Chapter 5](#). This chapter emphasizes the aspects related to leveraging pretrained models, including the non-interactive mode of the architecture.

When addressing fine-grained tasks in the low-data regime, it becomes crucial that the model has a sense for subtle differences, e.g., when two similar categories can only be distinguished by the color of a certain surface area of the object. Such low-level features are usually processed in the lower layers of deep neural networks and are replaced by higher-level, more abstract features in the upper layers [\[123, 124\]](#). In addition, pre-training is often done on datasets from the generic object detection domain, since this is the more widespread use case. As a result, pretrained models tend to develop inductive biases optimized for differentiating coarse but common categories such as chairs and tables. This can be especially unfavorable when target objects are visually very distinct from any of the classes in the dataset used for pre-training.

Given the challenges mentioned above, the developed system extends DINOv2, a model pre-trained with self-supervision and designed for representation learning, with lightweight components on top to recognize fine-grained categories. The self-supervised pre-training ([Chapter 2](#)) lets the model build a class-agnostic understanding of visual representations, which should be beneficial when recognizing rare or very specific items. For the recognition, these representations are then processed by a part-based inference pipeline, since this enforces finer granularity at the architectural level and is also more interpretable for human operators. The general concept of using parts for more interpretability is adapted from the ProtoPNets proposed by Chen *et al.* [\[105\]](#), while the interfaces are inspired, among others, by Bontempi *et al.* [\[106\]](#). At the top level, the inference structure based on global (category) and local (part) prototypes from Xue *et al.* [\[107\]](#) is comparable to this work’s method. However, they do not provide interactive interfaces, and although they use pretrained ViTs as backbones, full-model training is still required. This work incorporates self-supervised vision transformers that are leveraged for class-agnostic part recognition [\[38\]](#), so no specific training data with part annotations is needed. The DINOv2 ViT backbone is kept completely frozen to avoid costly training procedures, and only lightweight components are adapted each time to the task at hand. The priority was that these adaptations require low computational effort.

Parts of this section were previously published in [\[P3\]](#) and build on a supervised thesis [\[T6\]](#).

4.3.1 Target Scenarios

The system targets object recognition scenarios where object categories might only have very subtle differences. A simple example: two categories basically are the same object type, e.g., a screwdriver, but the handles of the crosstip screwdrivers are yellow, while the handles of the slotted screwdrivers are gray. Since only a few labeled examples per class are available, the model comes with capabilities to quickly recognize such differences without having to fine-tune. The objects are assumed to cover a significant area of the image such that even the parts are clearly visible. This assumption differentiates the approach from object detection, where the object can be depicted even with smaller scale in a cluttered scene. The model has been validated in scenarios with bird species and different types of power-drills, but should be transferable to other domains and categories, where all task-related objects have a similar overall structure and only the visual appearance of their parts determines the corresponding category. Note that, regarding the example with the screwdrivers, it would not be possible for the model to directly distinguish slotted from crosstip screwdrivers based on the geometric shape of their tips. Unfortunately, such shape based features are usually insufficiently represented in the extracted features of pretrained vision models.

4.3.2 Architecture and Method

The components of the developed architecture try to sequentially answer the following three questions:

- Where is the object?
- Which parts are visible?
- How similar are these parts to the parts of other categories?

Further, the components are arranged and designed to support interpretability and a human-guided adaptation, while keeping the large backbone frozen.

The complete architecture is depicted in Fig. 4.5. It describes the separate modules for foreground estimation, part separation, part similarity estimation, and part reweighting. As input serves the patch feature grid $\mathbf{F}$. This tensor contains the set of description vectors for all image patches extracted by the DINOv2 model [40], a vision transformer pretrained without supervision (Chapter 2). These description vectors are intermediate or output embeddings for all patches of the processed image, or a concatenation of those embeddings from multiple layers. Depending on the experiment and tasks different embeddings are used as description vectors. For instance, when the output tokens of the DINOv2 model in the *base* version are used, the description vector for a patch of 14×14 pixels is a 768-dimensional vector and the resulting $\mathbf{F} \in \mathbb{R}^{H_p \times W_p \times D}$ with $H_p \times W_p$ being the dimensions of the patchified image and $D = 768$ being the embedding dimension of the backbone (in this case the vision transformer used in the DINOv2 model). This matrix is used as the underlying feature representation for the input image for all the subsequent processing units, which will be described in the following. It is

important to mention that the steps are all category-specific, which means that they have to be performed in parallel for each category during inference.

Foreground Estimation

The goal of the foreground estimation module is to segment the area belonging to each foreground object depicted in each of the support images. During preparation this step can be simplified by creating pixel-level mask annotations beforehand for all the target objects on all support images. However, to avoid this effort and also comply with the default conditions in image classification tasks, no mask annotations and only image-level labels are assumed as given. Additionally, instead of using another specialized model which would increase the hardware requirements, the inherent foreground understanding of models in the self-Distillation with NO labels (DINO) family is leveraged by using attention rollout originally developed by [46] and adapted from [107]. This results in a segmentation map with the same resolution as the patchified input image. With this segmentation and the patch-wise extracted descriptors a linear Support Vector Machine (SVM) is trained for each category using all of the support images. During inference the SVM is used to predict whether a patch descriptor belongs to the foreground, i.e., the target object and replaces the rather demanding attention rollout according to

$$\hat{\mathbf{M}}_{\text{fg}} = f^{(1)}(\mathbf{F}) = \mathbf{F} \cdot \mathbf{W}^{(1)} + \mathbf{J} \cdot b^{(1)} \quad (4.1)$$

where the SVM weights are $\mathbf{W}^{(1)} \in \mathbb{R}^{D \times 1}$ and the bias $b^{(1)} \in \mathbb{R}$ is multiplied by a matrix of ones denoted as $\mathbf{J} \in \mathbb{R}^{H_p \times W_p}$. The clipped foreground map $\mathbf{M}_{\text{fg}} = \max(\min(\hat{\mathbf{M}}_{\text{fg}}, 1), 0)$ is used twice in the architecture: first in the foreground estimation module to weight the patch descriptors and then at the end to obtain the final score map.

Part Separation

To facilitate a fine-grained perception the object is split into separate parts during training and part-specific prototype vectors are created for each of them. During inference these prototypes can then be used to recognize and segment the corresponding part to enable the subsequent similarity estimation (Section 4.3.2). On one hand, the goal is that this works for any object type, even very rare ones, since the perception system should be as general as possible. On the other hand it should not require any part-oriented labeling, since part annotations are especially cumbersome. Trying to fulfill both requirements, the system exploits the inherent class-agnostic structural understanding of objects of the integrated large-scale pretrained DINOv2 architecture [40]. This specific capability was extensively examined by Amir *et al.* [38] for the predecessor model variant DINO [36]. They showed that the patch descriptors convey semantic information about which part of the object they belong to - even across different coarse categories. By a rather simple clustering procedure using the description vectors they were able to perform co-segmentation and part co-segmentation. For DINOv2 they showed the same capability [40], but instead of clustering, they visualized the first principal components.

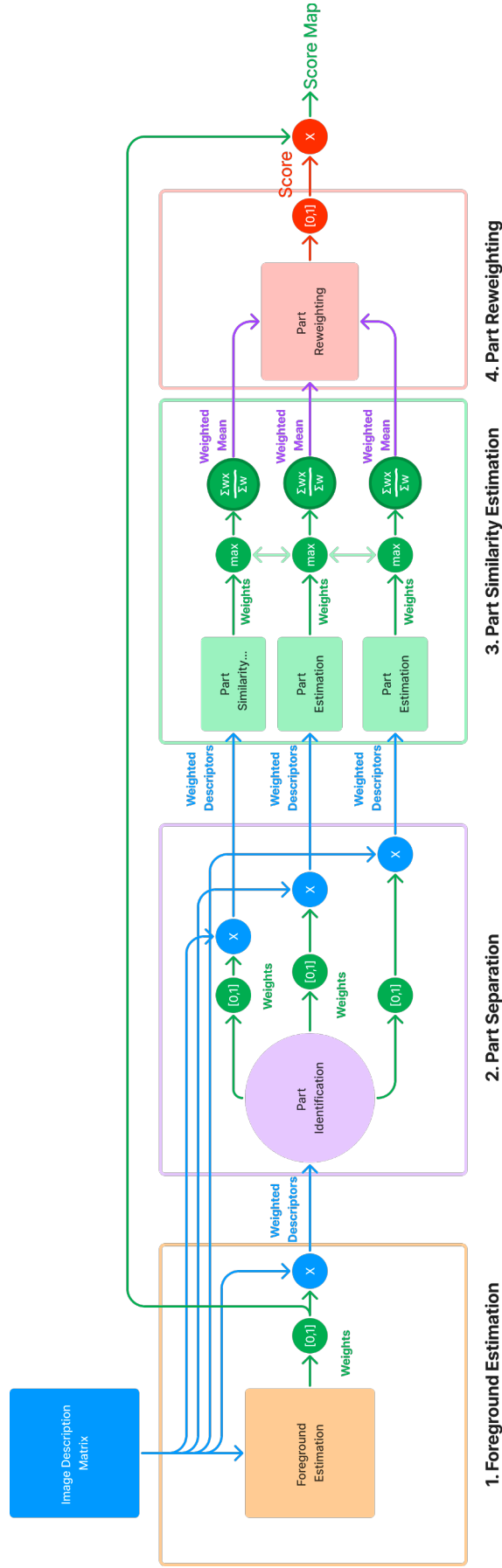


Figure 4.5: Overview of the part-based architecture for fine-grained object recognition. The image description matrix is created from the patch-wise embeddings of the pretrained vision transformer DINOv2. For each module a separate description matrix is prepared depending on the current configurations, e.g., sometimes this tensor only includes the output tokens for all patches and sometimes the concatenated ViT facet vectors across all layers. After segmenting the foreground object, it is split into separate parts, which are then compared with the parts of the known categories. For the description of parts prototype vectors are created during the preparation phase using the same pipeline up to the part separation. Copyright © 2024, IEEE [P3].

This class-agnostic structural understanding is used to separate any object into parts using a clustering method. Note that the resulting separated parts might be semantically coherent parts, such as a functional *handle* or anatomical *ear*. However, this is not guaranteed. The idea is, during inference, the object parts will be separated in the same way despite potential changes regarding the viewpoint. This enables a comparison with the same parts that were learned during the preparation phase. Parts are not separated on a pixel-level but rather softly defined by a part segmentation map similar to the foreground segmentation map.

The clustering method is K-means and usually $N_p < 10$. The larger N_p is chosen the smaller the parts get, which weakens robustness. On the contrary, with a smaller N_p , the parts get larger and the sensitivity regarding fine-grained differences is reduced. This makes N_p a category-specific hyperparameter. For part separation $\mathbf{F}_O \in \mathbb{R}^{H_p \times W_p \times C_O}$ is used, which consists of the output-layer features as patch descriptors to operate with the highest level of abstracted features. Here C_O represents the dimensions of the output-layer features. The final prototype for each part is the l2-normalized mean of the corresponding cluster. To ensure that each obtained cluster is assigned to the correct part prototype for comparison, the dot-product similarity between the two is used according to Eq. (4.2).

$$\hat{\mathbf{M}}_{\text{part}} = f^{(2)}(\mathbf{F}_O) = \mathbf{F}_O \cdot \mathbf{W}^{(2)} \quad (4.2)$$

Where $\mathbf{W}^{(2)} \in \mathbb{R}^{C_O \times N_p}$ contains the trained cluster centers (i.e., the part prototypes) as rows. This leads to an N_p -dimensional matrix which is multiplied with the foreground map from Eq. (4.1)

$$\mathbf{M}_{\text{part}} = \max(\min(\hat{\mathbf{M}}_{\text{part}}, 1), 0) \odot \mathbf{M}_{\text{fg}}. \quad (4.3)$$

where the resulting $\mathbf{M}_{\text{part}} \in \mathbb{R}^{H_p \times W_p \times N_p}$ contains the N_p clipped maps, each estimating the presence of the corresponding part in the patchified image.

Part Similarity

This step finally measures the similarity of the segmented parts with the ones that were learned from the support images during the preparation phase and is inspired by part-based prototypical networks such as ProtoPNet [105]. For comparing the similarity, the feature-rich part prototypes created by concatenation of ViT facets across all layers are used. The semantic similarity was the focus of the previous components, now the visual similarity becomes more relevant. Due to its increased depth it also takes attributes such as color and texture into account, which are processed on lower layers in ViTs.

During the preparation phase, using the part separation step, positive and negative training data samples in the form of patch-wise description vectors are collected for each part. This data is used to train a second linear SVM. During inference it predicts whether a patch belongs to the correct category or not using the multi-layer feature vectors. The step can be formalized

with the following equations

$$\hat{\mathbf{M}}_{\text{sim}} = f^{(3)}(\mathbf{F}) = \mathbf{F} \cdot \mathbf{W}^{(3)} + \mathbf{J} \cdot \mathbf{b}^{(3)} \quad (4.4)$$

$$\mathbf{M}_{\text{sim}} = \max(\min(\hat{\mathbf{M}}_{\text{sim}}, 1), 0) \odot \mathbf{M}_{\text{part}} \quad (4.5)$$

with the weights $\mathbf{W}^{(3)} \in \mathbb{R}^{D_F \times N_p}$ and the part-specific bias $\mathbf{b}^{(3)} \in \mathbb{R}^{N_p}$. Due to the element-wise product with the part separation maps $\mathbf{M}_{\text{part}}$, the focus on the correct part-related area is ensured. To only assign a patch to one part, the elements in $\mathbf{M}_{\text{sim}}$ along the N_p dimension that are non-maximal are set to zero. This means that only the part with the highest similarity score wins the assignment and the value is kept for the next steps. Now each patch belongs to one part and has a certain similarity with the multi-layer part prototype. As the final step the part score is summarized for the image and normalized with respect to the corresponding part segmentation maps $\mathbf{M}_{\text{part}}$ using

$$\hat{s}_p = \frac{\sum_{i \in H_p, j \in W_p} \tilde{\mathbf{M}}_{\text{sim}, i, j, p}}{\sum_{i \in H_p, j \in W_p} \mathbf{M}_{\text{part}, i, j, p}} \quad (4.6)$$

which calculates a similarity score for each part so that $[\hat{s}_1; \dots; \hat{s}_{N_p}]^T = \hat{\mathbf{s}} \in \mathbb{R}^{N_p}$ is obtained, with components in the range $[0, 1]$.

Final Prediction Score

As depicted in Fig. 4.5 the final object score is based on the part scores determined in the previous step. In the basic version the mean of all part scores $\hat{s}_p$ is the final score. In Chapter 5 a variant is examined where the operator has the option to vary the influence of each part to the final score. This enables ignoring certain parts for the final score, indirectly telling the model that the visual appearance of the corresponding part is not relevant for recognizing the specific category.

Concept Bank

The concept bank is a component that targets the semantic interpretation of the part prototypes and will be presented in detail in Chapter 5 since it is only relevant for the interactive use case scenario of this perception model.

4.3.3 Model Preparation

To prepare the model for operation, the user has the option to select between multiple variants of the DINO model. The larger models require more memory and computational power. However, their embeddings are superior and also have more dimensions. The advantage is

that the DINO models all use a vision transformer, so it can be easily exchanged without major architectural changes. But the preparation needs to be restarted from the beginning if the model is changed.

The number of clusters for the K-means clustering can alter performance depending on the object types, so the operator might want to test other values than the default value. The most effort is needed to train the SVMs. This can also be completely abstracted from the user since there are no hyperparameters involved that need adjustments each time there is a new task. While the system can process an arbitrary number of support images, this directly scales the computational effort for the SVM training. In general, all components on top of the DINO model are lightweight and linear, which reduces the preparation effort.

4.3.4 Inference

During inference the previously presented steps are performed sequentially for one category, and in parallel for all other categories that are part of the task at hand. Depending on the number of categories the hardware limits might be exceeded. The final category for each image is obtained by using the one with the highest final object score for the image-level prediction. A minimum threshold for predictions is currently not feasible, due to the properties of how the scores are calculated.

If the hardware is limited, the most relevant factors are the size variant of the DINO model and the target size of the input images. Unfortunately, both factors can have noticeable impact on the recognition performance. Further, the multi-layer patch descriptors require to create model hooks to extract ViT facets from intermediate layers. This slightly slows down the default inference time using the DINO models.

In case, new object categories need to be added, only the new corresponding support images need to be processed. For the previous ones, the image matrices can always be stored and reused for re-doing the preparation with the newly added categories.

4.4 Part-Enriched Prototypes for Fine-Grained Few-Shot Object Detection with Frozen Models

In this section, the fine-grained recognition task (Section 4.3) is addressed in an object detection setting. The developed system targets few-shot object detection, for which recent models have shown promising performance [88] without requiring fine-tuning on the support set. While the general concept of the system was introduced in [P2] and was originally designed for an interactive learning mode (Section 5.2), in this section, it is adapted to a non-interactive mode and used as a baseline for the later evaluation (Section 6.1.4) without claiming novelty.

When novel classes share many features or belong to a common super-class, it is challenging to keep them apart, especially when the models are not fine-tuned on the task-specific data. For such scenarios the system provides an alternative approach by incorporating prototypes on different granularity levels, and is inspired by fine-grained few-shot object recognition methods [105, 107, P3]. Prototypes therefore not only exist for the category as a whole, but also at a sub-level for characteristic visual attributes such as parts or logos. All prototypes are specific to each category. However, due to the general pretraining of the used vision transformer backbones, category-level prototypes for visually similar objects tend to be similar as well. To refine the initial predictions, the system also recognizes and evaluates the additional part-level prototypes, which allows it to distinguish even objects that look almost identical.

As mentioned before, the architecture is a non-interactive variant of the one presented in Section 5.2 and extends the high-performing two-stage Few-Shot Object Detection (FSOD) model Detect Everything Vision Transformer (DE-ViT) proposed by Zhang *et al.* [88], which relies entirely on class-level prototypes. Fine-grained category recognition is enabled by adding several specialized components and processing steps. The goal of these architectural adjustments is to recover the detailed information about categories that is lost when a single prototype is created for a whole category. This is done by representing the most characteristic parts with fine-grained prototypes. The added components are lightweight and sit on top of the existing model. The capabilities of the original DE-ViT model therefore remain available, which makes the model usable in both scenarios, generic and fine-grained few-shot object detection. The part-based inference also provides transparency and enables controllability. Chapter 5 shows how these properties can be leveraged for interactive human-in-the-loop few-shot object detection scenarios.

The advantage of the adapted DE-ViT model [88] is its fine-tuning-free preparation phase, which is unusual given that most FSOD approaches rely on fine-tuning on the support set. The category-level prototypes are obtained from the support set using forward passes only, without any gradient calculations. Other fine-tuning-free FSOD methods exist [80, 81, 85, 86, 87]. Nevertheless, DE-ViT is currently one of the best-performing FSOD models. In a direct comparison, it was outperformed by a three-stage detector proposed by Eisenbach *et al.* [125], which also does not require fine-tuning and behaves similarly to deep template matching methods. However, that evaluation was done in an object re-identification setting, where

the support set consists of images of the same item captured from different perspectives. This is not the primary use case of DE-ViT. The architecture presented in this section aims to add optional fine-grained perception to the generic category-level detection capabilities of DE-ViT. This extends its ability to handle task scenarios with low intra-class variability and high inter-class similarity, such as object re-identification, when required.

4.4.1 Target Scenarios

The system targets robotic scenarios in which the perception capabilities must be quickly adapted to novel objects that have to be detected in a potentially cluttered environment. Only limited annotated data is assumed available for these objects, and a fine-tuning step is infeasible or undesirable, e.g., due to limited computational power of the onboard hardware or a shortage of time. While the previous section addresses object recognition, where the object is assumed to cover a large area of the query image, here the objects can appear smaller and can be located anywhere in the image. Naturally, fine-grained features become less present the smaller the area the object covers. If the objects are too small, characteristic features tend to vanish in the embeddings of the pretrained backbone, which makes subsequent fine-grained inference nearly impossible.

As mentioned in the previous section, generic and fine-grained few-shot object detection are possible at the same time with this system. Fine-grained detection is, however, limited to categories that can be distinguished by visual features and not by purely functional differences. These visual features must be locatable in a constrained region on the object's surface and must also have a characteristic representation in the embeddings of the backbone. Note that in the current version the model does not take the relative position of these features into account. If the positional feature arrangement is the only difference between two categories, they *appear* identical to the model and the FRM.

4.4.2 Architecture and Method

Following the general approach in this chapter, the backbone of the architecture is a pretrained backbone. More precisely, it is the pretrained vision transformer DINOv2 [40] (Chapter 2). It is kept completely frozen, so even during preparation no fine-tuning step changes any weights. In addition, to derive potential object locations, a RPN runs in parallel. The overall architecture therefore belongs to the family of two-stage detectors (Chapter 2). Figure 4.6 depicts the complete architecture and visualizes the information flow between the relevant components during operation and inference. The RPN is also pretrained, but not in an unsupervised manner. Instead, it is trained with supervision on a significantly smaller dataset. In theory this is a weak spot, because if the novel objects for the task at hand have a significantly different appearance or *objectness*, at least from the perspective of the RPN, then no box proposals might be generated for them. Without proposals, the rest of the architecture will not process

these image areas any further and is effectively *blind*. In practice, RPNs often develop a generalized capability for all sorts of objects, so that even novel objects are recognized and box proposals are predicted.

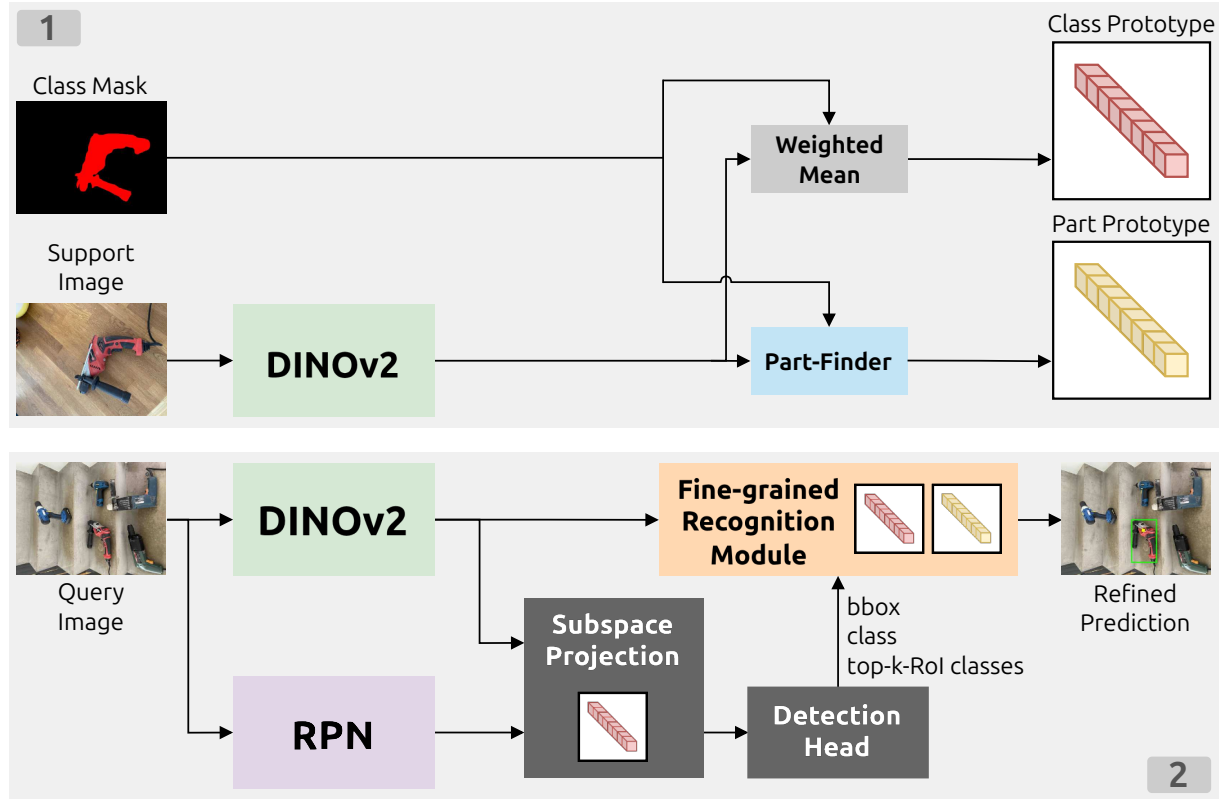


Figure 4.6: The Detect Everything fine-grained Vision Transformer (DE-fine-ViT) model based on the DE-ViT components [88] (dark gray). Through extending it with additional part prototypes fine-grained few-shot object detection is enabled. In the preparation phase ❶ the support images are processed by the DINO backbone and the non-interactive Part-Finder module (blue) discovers distinctive part prototypes. During the inference ❷ a part-based prediction re-evaluation is performed (orange), which takes all class prototypes as well as their part prototypes into account to resolve confusions of similarly looking categories. The illustration depicts only a single class, part, and support example. However, in practice, multiple of each are typically considered.

It is quite common for two-stage detectors that the RPN and the detection network share their backbone. In the underlying architecture, adapted from the DE-ViT model [88], this is not the case. A promising research direction for future versions is to generate box proposals from the DINOv2 embeddings as well. During the operation and inference phase, the features extracted by DINOv2 are aligned and processed by a simplified dot-product classifier that uses the class-level prototypes. This classifier determines the top- k_{det} classes for which the detection head predicts a refined bounding box, the underlying mask, and a class score. The prediction with the highest class score is then passed to the Fine-grained Recognition Module (FRM) together with the list of the top- k_{ROI} classes determined by the classifier in the beginning. The module then estimates the presence of characteristic parts in the image features constrained by the corresponding bounding box. To reduce the complexity of the fine-grained recognition problem, only the part prototypes of the most similar categories are taken into account. The

details of this procedure are described later in this section. The module finally outputs the (potentially updated) fine-grained detection.

During the preparation phase, class and part prototypes are created based on the annotated data. Since on-the-fly part-based annotation can be quite tedious, the interactive use case of this model integrates a class-agnostic pretrained segmentation model that the operator can trigger to create pixel-level labels with a few clicks. This procedure is described in more detail in [Chapter 5](#).

Multi-Granularity of Prototypes

The DE-ViT architecture proposed by Zhang *et al.* [88] focuses on general few-shot object detection and incorporates class prototypes based on DINOv2 [40] embeddings. To improve fine-grained capabilities, part prototypes are introduced.

The class prototypes, which are created according to

$$\mathbf{p}_{c,i}^{(\text{cls})} = \frac{1}{\sum_{j=1}^{H_p W_p} \alpha_{c,j}^{(\text{tkn})}} \sum_{j=1}^{H_p W_p} \mathbf{f}_{i,j} \cdot \alpha_{c,j}^{(\text{tkn})} \quad (4.7)$$

with $H_p W_p$ as the total number of image patches, $\mathbf{f}_{i,j} \in \mathbb{R}^D$ being the feature vector for the patch at location (i,j) , and the scalar factor $\alpha_{c,j}^{(\text{tkn})} \in [0,1]$ which represents the fraction of the patch covered by the class annotation mask or bounding box, which represents the default case. The global class prototype is then the average of all the prototypes calculated for each of the support images for the corresponding class

$$\mathbf{p}_c^{(\text{cls})} = \frac{1}{K} \sum_{i=1}^K \frac{\mathbf{p}_{c,i}^{(\text{cls})}}{\|\mathbf{p}_{c,i}^{(\text{cls})}\|_2} \quad (4.8)$$

with K as the total number of support images available for category c .

Part prototypes, in contrast, are obtained automatically as described below. An interactive variant of this procedure is presented in [Section 5.2](#). The corresponding strengths and weaknesses are compared in the evaluation ([Section 6.1.4](#)).

Automatic Part-Prototype Discovery

Manually annotating part masks for every novel class is not always feasible. The following describes an automatic method for the DE-fine-ViT architecture [P2] that discovers part prototypes on its own. It only uses the available class-level bounding box annotations, which are already needed for [Eq. \(4.7\)](#). The procedure consists of three stages: (i) determine potential part candidates, (ii) score and compare properties of those candidates, and (iii) final selection. The output is a set of part prototypes $\{\mathbf{p}_{c,j}^{(\text{part})}\}$ with optionally associated importance weights

$\{w_{c,j}\}$ that fit into the weighted re-evaluation of [Algorithm 1](#), which was originally developed for the interactive approach ([Section 5.2](#)).

Creation of Part Candidates

For every support box b of class c , the DINOv2 patch features $\mathbf{F}_b \in \mathbb{R}^{N_b \times D}$ are partitioned into N_{cand} candidates using k -medoids [126] with cosine distance. Each candidate $\hat{\mathbf{p}}_{b,k}^{(\text{part})}$ is taken to be the medoid of its cluster, and the set $\mathcal{C}_{b,k}$ of patches assigned to it is retained. The union of candidates extracted from all support boxes of class c forms a class-level pool $\mathcal{P}_c^{\text{cand}}$ from which the final part prototypes are selected.

Part Candidate Scoring

Each candidate $\hat{\mathbf{p}}^{(\text{part})} \in \mathcal{P}_c^{\text{cand}}$ is rated by three normalized scores in $[0,1]$ that assess certain intuitive properties a part prototype should exhibit:

- **Recurrence** $s_{\text{rec}}(\hat{\mathbf{p}}^{(\text{part})})$: the fraction of *other* support boxes of class c that contain at least one patch with $\cos(\cdot, \hat{\mathbf{p}}^{(\text{part})}) > \tau_{\text{rec}}$. Assumption: recurring parts are likely characteristic of the class rather than instance-specific artifacts.
- **Discriminativeness** $s_{\text{disc}}(\hat{\mathbf{p}}^{(\text{part})})$: the distance to part candidates of other classes, weighted by how *confusable* those classes are with c at the global level:

$$s_{\text{disc}}(\hat{\mathbf{p}}^{(\text{part})}) = \sum_{c' \neq c} \omega_{c,c'} \left(1 - \max_{\hat{\mathbf{p}}'^{(\text{part})} \in \mathcal{P}_{c'}^{\text{cand}}} \max(0, \cos(\hat{\mathbf{p}}^{(\text{part})}, \hat{\mathbf{p}}'^{(\text{part})})) \right) \quad (4.9)$$

$$\omega_{c,c'} = \frac{\max(0, \cos(\mathbf{p}_c^{(\text{cls})}, \mathbf{p}_{c'}^{(\text{cls})}))}{\sum_{c'' \neq c} \max(0, \cos(\mathbf{p}_c^{(\text{cls})}, \mathbf{p}_{c''}^{(\text{cls})}))}. \quad (4.10)$$

Assumption: selecting a part shared with a visually distant class is neither harmful nor beneficial ($\omega_{c,c'}$ is low), whereas one shared with a visually similar class can be harmful. Therefore, only parts relevant for similar classes are desired ($\omega_{c,c'}$ is high), and they must be distinct enough to help resolve misclassifications in the fine-grained stage.

- **Spatial compactness** $s_{\text{cmp}}(\hat{\mathbf{p}}_{b,k}^{(\text{part})})$: the spatial concentration in the image space of the patches in the candidate's own cluster $\mathcal{C}_{b,k}$. Let $\mathbf{x}_{b,j}$ with its components in $[0,1]$ denote the normalized center location in image coordinates of patch j in support box b , and

$$\bar{\mathbf{x}}_{b,k} = \frac{1}{|\mathcal{C}_{b,k}|} \sum_{j \in \mathcal{C}_{b,k}} \mathbf{x}_{b,j} \quad (4.11)$$

be the spatial center of the cluster. Then the compactness score becomes

$$s_{\text{cmp}}(\hat{\mathbf{p}}_{b,k}^{(\text{part})}) = \frac{1}{1 + \frac{1}{|\mathcal{C}_{b,k}|} \sum_{j \in \mathcal{C}_{b,k}} \|\mathbf{x}_{b,j} - \bar{\mathbf{x}}_{b,k}\|_2}. \quad (4.12)$$

Compact candidates correspond to localized part hypotheses. Assumption: clusters whose patches are scattered across the bounding box are less interpretable and convey less semantic information.

The three scores are combined via multiplication,

$$s(\hat{\mathbf{p}}^{(\text{part})}) = s_{\text{rec}}(\hat{\mathbf{p}}^{(\text{part})}) \cdot s_{\text{disc}}(\hat{\mathbf{p}}^{(\text{part})}) \cdot s_{\text{cmp}}(\hat{\mathbf{p}}^{(\text{part})}), \quad (4.13)$$

so that a candidate must satisfy all three criteria simultaneously to be retained.

Selection

Near-duplicates within $\mathcal{P}_c^{\text{cand}}$ are removed by a greedy non-maximum-suppression on cosine similarity with threshold τ_{dedup} , which keeps the higher-scoring representative. The top- N_{parts} surviving candidates form the part prototypes $\{\mathbf{p}_{c,j}^{(\text{part})}\}$ of class c , and their scores can be used directly as importance weights. Both can then be processed by the weighted re-evaluation in [Algorithm 1](#).

Refining Predictions

The developed DE-fine-ViT model builds on the general detection capabilities of the DE-ViT model [88] and extends it with the FRM ([Fig. 4.6](#)). The pseudocode in [Algorithm 1](#) gives a high-level description of the re-evaluation procedure within the module. The intuition behind it is that, based on the parts of a category and their presence in the support images, views are defined, and each view is associated with a specific subset of parts that should be visible. These views are then used as a reference during inference on a given query image. For this, the similarity scores between the image and each view of the top- k_{det} categories are estimated using the average similarity of the class-level prototype in the area defined by the bounding box, together with the maximum similarities of the part prototypes that are expected to be visible for the corresponding view.

The category of the best matching view-specific part combination becomes the final predicted category and replaces the previously predicted one. Neither the bounding box nor the mask is changed. In addition, the prediction is extended by the predicted locations of the view-specific parts within the bounding box. These pixel-level part locations can be used for downstream tasks. For instance, they can serve as seeds for generating grasp trajectories for the robot.

Algorithm 1 Fine-Grained Part-Based Prediction Re-evaluation

Require: Initial DE-ViT detections with top- k_{RoI} candidate class sets $\mathcal{T}_i^{(k)}$

Require: Class prototype bank $\mathbf{P}_{\text{cls}}$, part prototypes $\{\mathbf{p}_{c,j}^{(\text{part})}\}$ organized by view

Ensure: Re-evaluated detections with refined class labels and similarity scores

- 1: **for** each detection i and its candidate set $\mathcal{T}_i^{(k)}$ **do**
- 2: Extract patch features $\mathbf{F}_b$ from the bounding box b_i
- 3: **for** each candidate class $c \in \mathcal{T}_i^{(k)}$ **do**
- 4: Compute cosine similarity between $\mathbf{F}_b$ and class prototype $\mathbf{p}_c^{(\text{cls})}$
- 5: **if** similarity below threshold τ_{conf} **then**
- 6: Reject candidate
- 7: **else if** no part prototypes exist for class c **then**
- 8: Score using class-level similarity only
- 9: **else**
- 10: **for** each view v of class c **do**
- 11: Match all part prototypes $\mathbf{p}_{c,v,j}^{(\text{part})}$ against $\mathbf{F}_b$
- 12: Aggregate part similarities as weighted average with weights $w_{c,v,j}$
- 13: **end for**
- 14: Select best-scoring view
- 15: **end if**
- 16: **end for**
- 17: Keep candidate with highest final similarity
- 18: **end for**
- 19: **return** refined detections

4.4.3 Model Preparation

Preparing the model does not require any fine-tuning. The weights for the pretrained models, such as DINOv2, the RPN, and the DE-ViT detection head, are publicly available [127]. Once these models are obtained and ready to use, the class and part prototypes are created by performing a single inference pass on all support images with the DINOv2 model. After the inference pass, a lightweight post-processing step on the image-level patch embeddings computes all prototypes for the given class annotations. Creating these annotations is usually the most time-consuming preparation step, but they are necessary. They also lead to better performance when provided as pixel-level binary masks rather than bounding boxes. The prototypes are then combined before being loaded by the DE-fine-ViT model for the operation phase. While the prototypes are obtained, the corresponding *views* are also created automatically. Each unique combination of category-specific parts in the support images is treated as a different view. Finally, the similarity of class and part prototypes is assessed based on the list of views. If high similarities are found, the model is likely to confuse the corresponding categories. To prevent this, the proposed interactive variant of the system already takes measures during the annotation phase that involve the human operator (Chapter 5). The automatic part discovery method, in contrast, relies on the effectiveness of the scores to select the most suitable parts for the task at hand. The preparation effort increases with the number of categories.

4.4.4 Inference

During inference, the model takes each box proposal from the RPN and forwards it to the detection head of the underlying DE-ViT model. The detection head only considers the top- k_{det} classes for each box-specific inference step. This, together with the fact that the RPN uses its own backbone instead of the DE-ViT backbone DINOv2, leads to relatively high GPU memory demands. At no point are backpropagation and gradients required, which would further increase these demands.

The model has several relevant thresholds that need to be set and that can be used to adjust the detection behavior and performance. Since no fine-tuning is involved, these can be re-adjusted even during operation if malfunctioning is observed.

4.5 Summary and Conclusion

The methods presented in this chapter are developed to enable object-centric visual perception in robotic systems under resource constraints by leveraging pretrained models and by minimizing or even completely avoiding task-specific training. Rather than adapting model parameters through costly optimization procedures, the methods either rely on short fine-tuning phases or restructure and exploit the rich feature representations already embedded in large-scale pretrained vision models.

One methodological focus lies on the use of large-scale pretrained backbones, with and without supervision, whose feature spaces are repurposed for downstream tasks such as segmentation, classification, and detection. Instead of gradient-based adaptation, lightweight mechanisms such as prototype-based representations, similarity computations, and modular post-processing are used to specialize these general-purpose embeddings to new tasks. This approach reduces the dependence on large annotated datasets and removes the need for computationally intensive training cycles.

Multiple complementary strategies are developed. For segmentation tasks, pretrained models are used to extract attention and feature structures that can be translated directly into object masks with minimal supervision. For more complex perception scenarios involving high intra-class variability, additional sensor modalities such as depth are integrated through fusion pipelines, improving robustness without increasing training complexity when geometric properties help to distinguish categories. For recognition and detection tasks, few-shot learning principles are built on top of prototype-based formulations, where categories are represented by compact descriptors derived from a small number of annotated examples.

A common feature across several methods is the introduction of multi-granularity representations. Instead of relying only on global class-level prototypes, the approaches incorporate finer structural information through part-level prototypes. This allows the system to distinguish between visually similar or nearly identical categories by focusing on characteristic local features. The resulting architectures keep their general recognition capabilities while also covering fine-grained tasks, bridging the gap between coarse and detailed perception without requiring separate models. Preparation is mostly reduced to a forward-pass extraction of feature embeddings and the construction of prototype representations, which results in a linear and predictable scaling with the number of categories. This design makes the methods particularly suitable for deployment on resource-limited robotic platforms.

Relying on sufficiently expressive representations comes with inherent limitations. Since no fine-tuning is performed, performance depends heavily on the quality and generalization capabilities of the underlying pretrained models. In addition, the effectiveness of fine-grained recognition is constrained by the visibility and representability of discriminative features in the embedding space. Small object sizes, occlusions, and purely functional differences between categories remain challenging.

In summary, the methods in this chapter establish a practically oriented framework for resource-efficient robotic perception. Training-heavy pipelines are replaced with prototype-based reasoning and modular extensions, which shifts the focus to the underlying representations and inference mechanisms used to exploit pretrained models in real-world robotic applications.

5 Interaction-Based Few-Shot Perception

For vision tasks, it is the training data that *defines* what the model has to learn. However, tasks can usually only be specified implicitly through data. In contrast, the human operator knows the *real* task, or at least has access to relevant context knowledge from sources other than data. The resulting information gap to the *real* task is usually minimized by adjusting the training data, the learning procedure, and the architecture type to obtain the desired perception model. However, due to the practical imperfection of the training dataset, the model faces several difficulties when learning the correct task (Section 3.1.1). The effort required to increase the amount of labeled data and compensate for this through more extensive training can be infeasible in certain scenarios. While in Chapter 4 pretrained models are leveraged and trusted to apply the right inductive biases, this chapter presents methods that follow an interactive learning paradigm: the human operator is incorporated into the preparation and inference phase (Fig. 5.1) to reduce the task specification gap described above [57]. Their role is to assess, correct, and guide the model through appropriate interfaces, so that the right information about the task is extracted from the few training samples.

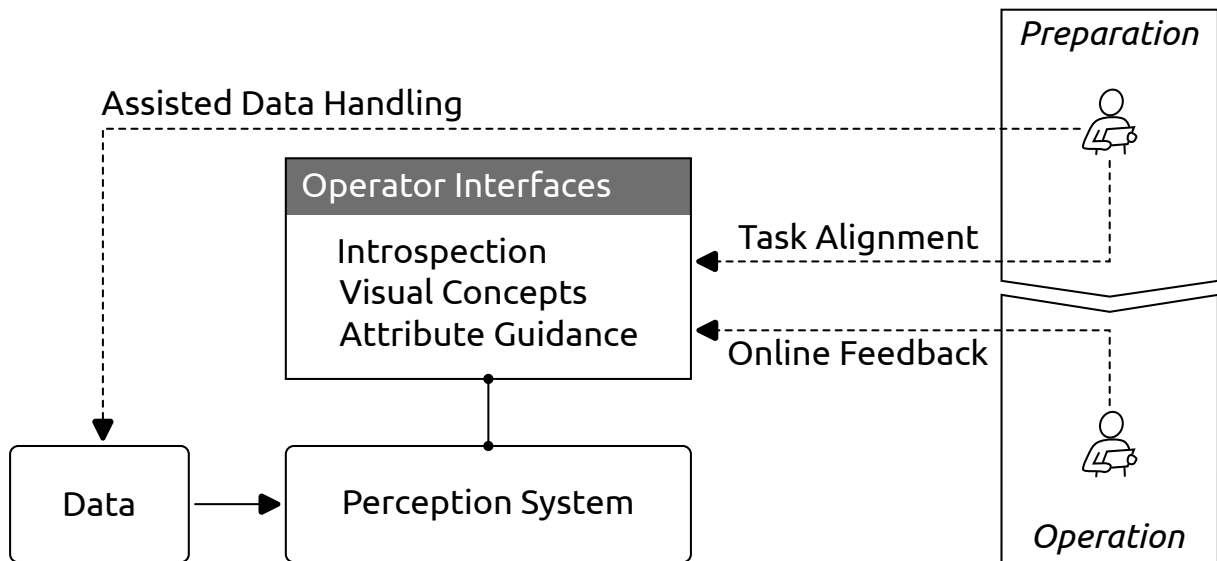


Figure 5.1: The schematic shows the connections between the high-level components of the proposed concept that is covered in this chapter. The main challenge addressed by the concept is the potential misalignment between the *real* task the operator has in mind and the model’s understanding of the same task, which is derived from the few available data samples. By providing human interfaces for the proposed vision architectures, the operator can improve the task alignment during the preparation phase and can still give online feedback during the operation phase to adjust the system’s behavior. Assistive tools for data handling are integrated to increase the speed and reduce the required proficiency.

5.1 Interactive Semantic Modification of Part Prototypes

This section describes the interactive variant of the architecture presented in Section 4.3. The target task is few-shot recognition for fine-grained scenarios. It shifts the role of part prototypes from being static and defined only by the training data to modifiable representations. For the modification, a concept bank is integrated as a translator between human-understandable semantic concepts and the embedding space of the high-dimensional prototypes. With this, the inference process can be analyzed for dominant semantic concepts associated with a certain object part, and it can also be altered without any retraining of the backbone or the general pipeline. For altering, explicit interfaces on the part level are integrated that first serve to inspect the influence of certain semantic concepts. This influence can then be modulated, increasing or decreasing the relevance of a concept for recognizing the part. The semantic interpretability is modeled by using concept activation vectors introduced by Kim *et al.* [50] and the idea of using them for a concept bank is inspired by the Discover and Cure (DISC) framework proposed by Wu *et al.* [51]. Within the DISC framework, the concept bank is used to discover spurious features *during* the training process of image classifiers. The framework then mitigates their use by the classifier with concept-aware data augmentation to obtain a better balanced dataset for the next training iteration.

Note that in the interactive mode, this approach still keeps the complete Vision Transformer (ViT) backbone frozen and provides post-hoc adaptability via user interfaces without any gradient-based fine-tuning. This way, the model’s decision process can be adjusted after the

Table 5.1: Comparison of the non-interactive and interactive VINTER modes.

Aspect	Non-interactive Mode	Interactive Mode
Backbone Adaptation	Frozen pretrained ViT	<i>identical</i>
Feature Representation	ViT patch descriptors	<i>identical</i>
Decision Structure	Multi-stage pipeline	<i>identical</i> , components modifiable via interfaces
Foreground Estimation	Learned once during preparation	<i>identical</i>
Part Separation	Semantic parts defined by clustering	<i>identical</i>
Part Similarity	Part similarity learned from data	Initial state is identical, prototypes can be analyzed & adjusted
Part Aggregation	Fixed aggregation	User-adjustable part weights to emphasize or suppress parts
Interpretability	Implicit via part-wise scores and association maps	Explicit via concept bank and CAVs
Options for Adaptation	Additional data and retraining	Interactive interfaces for prototypes and part aggregation
Computational Overhead	<i>Baseline</i>	Overhead due to management of concept bank

initial few-shot training (Section 4.3) without the need to acquire more or different data (Table 5.1).

Parts of this section were previously published in [P3] and build on a supervised thesis [T6].

5.1.1 Target Scenarios

With the interactive aspect, this approach extends the functionality of the base model (Section 4.3) and its target scenarios described in Section 4.3.1. Even if the initial training data is insufficient, the operator can use the interfaces to adapt the model's inference.

Further, the inspection tools based on interpretability methods are suitable for failure analysis, to better understand which part causes a misclassification and why. Based on these insights, misleading feature correlations can be mitigated and the relevant visual characteristics of a category can be specified more clearly.

5.1.2 Architecture and Method

The general architecture is identical to the one described in Section 4.3 for fine-grained few-shot object recognition. To provide the full interactive mode, it is extended with a connected concept bank (Chapter 2) that can translate features in the embedding space into semantic concepts understandable by humans. In total, there are three interfaces that can be used to change the decision process of the perception model:

- part reweighting
- prototype characterization
- prototype modification.

They are described in detail in the following. For an in-depth explanation of the other components, refer to Section 4.3. Note that these interfaces only target the layers built on top of the pretrained backbone and are optional. After the initial adaptation to the novel classes as described in Section 4.3, the model is already operational. However, if the operator has access to validation data, the evaluation results can be used to adjust the model and improve its fit to the task at hand.

Part Reweighting

This component is located at the end of the architecture. Before the part reweighting, the part similarity estimation module produces a part score vector that estimates in the range $[0, 1]$ how visually similar the parts found in the input image are to the ones learned for a given

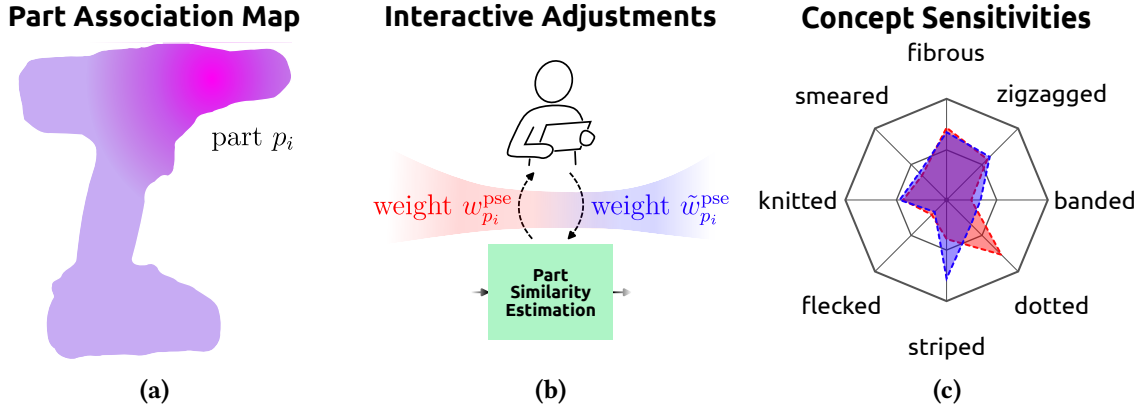


Figure 5.2: The VINTER model (Section 4.3) is trained on a support set and learns to separate different semantic parts of an object, e.g., part p_i focuses on the tip of the power drill (a). At the same time, a subsequent module, the part similarity estimation, derives a multi-layer prototype representing the visual appearance of each of those parts. However, since there are only few samples, it might pick up wrongly correlated features and overestimate their relevance. Therefore, the operator can inspect the **current sensitivities** (c) of each part (here part p_i) for a given concept bank (here 8 textures). Desired adjustments can be made by modifying the weights $w_{p_i}^{pse}$ of the corresponding SVM for part similarity estimation using CAVs. The adjusted $\tilde{w}_{p_i}^{pse}$ (b) increases the relevance of a striped appearance, resulting in **new sensitivities** (c). Note that the relevance of other concepts is indirectly affected as well. A geometrical visualization of this modification is depicted in Fig. 5.3.

category. This vector is then used to calculate the final prediction score with

$$s^{(pr)} = \frac{1}{N_p} \sum_{p=1}^{N_p} \hat{s}_p \quad (5.1)$$

which weighs all N_p parts equally. However, the visual appearance of some parts might not be discriminative for the category required in the current task, or the other way around: some parts are only useful for distinguishing two similar categories. One use case would be that the model shall detect all power drills with an orange handle, but the color of the tip does not matter. To provide the operator with a way to communicate this, a weighted projection is defined instead of Eq. (5.1), which introduces part-specific weights w_p to scale the relevance of a part relative to the other parts. Higher weights w_p emphasize the presence of certain parts, and lower weights reduce their role in recognizing the corresponding category. The weighted projection

$$s^{(pr)} = \frac{1}{\sum_{p=1}^{N_p} w_p} \sum_{p=1}^{N_p} w_p \cdot \hat{s}_p \quad (5.2)$$

uses w_p as the adjustable weight for part p . These weights are class-specific, which is omitted in the notation to improve readability. The default value for all part weights is 1, which makes the final score equal to the mean in Eq. (5.1). The user can then adjust these values according to the task's requirements. A possible improvement would be to optimize these weights on the initial dataset for the novel classes before allowing the operator to adjust them.

Prototype Characterization

Before the prototypes can be modified, the operator needs to know which semantic concepts the model currently associates with each part. Wu *et al.* [51] introduced a concept bank $\mathcal{B}$ for similar reasons. The number of concepts in $\mathcal{B}$ is finite and bound to the semantic categories used to create the concept bank. The main idea is to assign meaning to directions inside the high-dimensional embedding space to increase the interpretability. The concept bank acts like a *dictionary* for all known directions and their corresponding semantic attributes. Note that a concept bank is typically created offline before any preparation phase, since it involves specific semantic datasets and several processing steps. In-depth details about concept banks can be found in Chapter 2. In the following, a shorter, high-level description is given to focus on the aspects relevant for the proposed interactive interfaces.

A concept is represented by a Concept Activation Vector (CAV) [50] (Chapter 2). To obtain them, linear classifiers are first trained using semantic categories from a semantic dataset such as Broden [128]. These classifiers take patch descriptors as input and predict whether the patch belongs to a specific semantic category. The semantic categories of the Broden dataset are colors, textures, materials, parts, objects, and scenes. For the use case of fine-grained few-shot recognition, only classifiers for classes belonging to colors and textures are trained on the corresponding patch descriptors obtained from the pretrained backbone DINOv2 [40]. In addition, a classifier is not trained on all classes. Instead, a subset of the 58 classes belonging to colors and textures is chosen, called concept set $\mathcal{A}$. The selection depends on the concepts the operator wants to have available during prototype analysis and in the subsequent prototype modification step. Wu *et al.* [51] only used the highest layer’s output to create concept banks. This work also considers other layers and different facets of the ViT backbone, since lower layers are more sensitive to low-level features such as colors and textures.

A linear concept predictor $f_{a_j} : \mathbb{R}^D \rightarrow [0, 1]$ for concept a_j in concept set $\mathcal{A}_i$

$$f_{a_j}(\mathbf{f}_{i,j}) = \text{clip}(\mathbf{w}_{a_j} \cdot \mathbf{f}_{i,j} + b_{a_j}, 0, 1) \quad (5.3)$$

uses D as the number of dimensions of the corresponding description vectors. Then the normalized vector

$$\mathbf{v}_{a_j} = \frac{\mathbf{w}_{a_j}}{\|\mathbf{w}_{a_j}\|_2} \quad (5.4)$$

can be seen as the CAV for concept a_j , since it would increase the prediction score if $\mathbf{f}_{i,j}$ is shifted in this direction. Then $\mathcal{V}_i^{\mathcal{A}} = \text{span}(\mathbf{v}_{a_1}, \dots, \mathbf{v}_{a_J})$ defines the subspace of linear combinations of all CAVs within the embedding space of the descriptors $\mathbf{f}_{i,j}$. By projecting $\mathbf{w}_p^{\text{pse,norm}}$ onto $\mathcal{V}_i^{\mathcal{A}}$

$$\text{proj}_{\mathcal{V}_i^{\mathcal{A}}}(\mathbf{w}_p^{\text{pse,norm}}) = \sum_j^{|\mathcal{A}|} (\mathbf{w}_p^{\text{pse,norm}} \cdot \mathbf{v}_{a_j}^\perp) \cdot \mathbf{v}_{a_j}^\perp \quad (5.5)$$

where $\{\mathbf{v}_{a_1}^\perp, \dots, \mathbf{v}_{a_r}^\perp\}$ is an orthonormal basis of $\mathcal{V}_i^A$, and measuring the length of the projection, it can be determined how sensitive the part classification is to each of the concept activation vectors of $\mathcal{V}_i^A$. If the length is 1, the classifier can be fully replaced by a linear combination of the CAVs. If the length is 0, the classifier lies outside $\mathcal{V}_i^A$ and is not sensitive to any of the obtained $\mathbf{v}_{a_j}$. The specific sensitivity for a CAV inside the linear part similarity estimation layer can be calculated using

$$\alpha_{i,j,p} = \mathbf{w}_p^{\text{pse}} \cdot \mathbf{v}_{a_j} \quad (5.6)$$

according to the sensitivity definition in [50], already simplified by taking the subsequent linear layer into account.

Kim *et al.* [50] note that CAVs can be created for any set of images, even if they are random and do not belong to any meaningful semantic concept. They therefore propose several statistical validation steps to ensure sufficient reliability and interpretability of the obtained CAVs.

Prototype Modification

In the previous section, the analysis of the part similarity estimation layer was explained. The result of this analysis is a concept set $\mathcal{A}_i$ and the associated concept predictors $\{(f_{a_j})\}_{j=1}^{|\mathcal{A}_i|}$. Further, the sensitivities of the part similarity estimation layer with respect to the concepts in $\mathcal{A}_i$ are given by the $\alpha_{i,j,p}$ values. While Kim *et al.* [50] treat these sensitivities as static model properties for interpretability purposes only, Wu *et al.* [51] already try to modify them implicitly through data augmentation and data selection strategies. Since the pipeline for few-shot object recognition presented in this work consists exclusively of linear layers, the sensitivity equation is rather straightforward. Therefore, in Eq. (5.6), changing the sensitivity is as simple as modifying the weights of the linear layer $\mathbf{w}_p^{\text{pse}}$ according to

$$\tilde{\mathbf{w}}_p^{\text{pse}} = \mathbf{w}_p^{\text{pse}} + \lambda \hat{\mathbf{v}}_{a_j} \quad (5.7)$$

with a factor λ that reaches a target sensitivity α^* if chosen as $\lambda = \alpha^* - \alpha_{i,j,p}$. Note that the concept activation vectors cannot be assumed to be orthogonal to each other. Only when they are orthogonal can one concept sensitivity be changed without indirectly modifying other sensitivities at the same time. In all other cases, there will be interference. A simplified visualization of the procedure is depicted in Fig. 5.3.

5.1.3 Model Preparation

Since the interactive architecture builds on top of the default version presented in Section 4.3, the same initial preparation steps are required. In addition, the concept banks need to be created based on semantic datasets. Existing ones, such as Broden [128], can be used. Creating custom concept banks that contain the necessary concepts for the task at hand would be rather

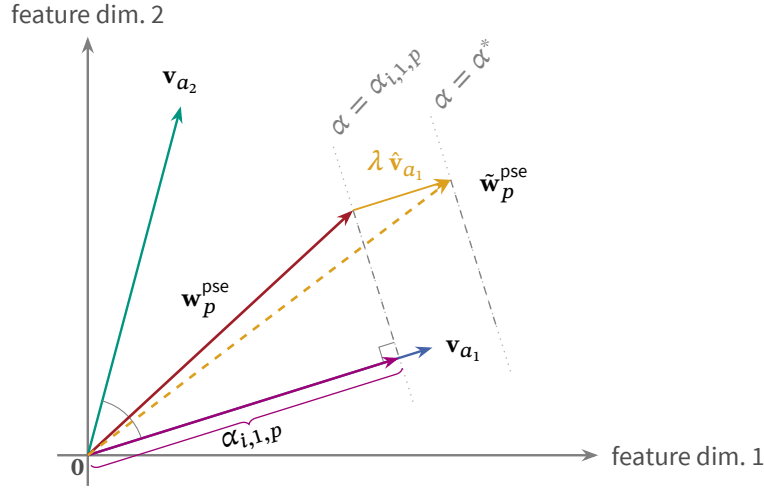


Figure 5.3: Simplified overview of the prototype modification. The sensitivity regarding concept a_1 is increased by $\lambda \hat{v}_{a_1}$. Since concept activation vectors (CAVs) are not necessarily orthogonal, this update influences the a_2 sensitivity as well.

time-consuming and would require special expertise. This would severely limit the practical applicability.

To create a concept bank, the backbone is used for gradient-free forward passes of all selected samples of the semantic concept dataset, in order to obtain the corresponding patch description vectors. This requires small amounts of GPU memory and sufficient free disk storage. The advantage is that the concept bank is task-agnostic and can be reused for other tasks.

5.1.4 Inference

The inference pipeline is identical to the default mode without interaction (Section 4.3), except that the weights of some model components might differ due to adjustments made by the operator to the part aggregation weights or the part similarity estimation components. The three interactive elements used for such adjustments, part reweighting, prototype characterization, and prototype modification, are mainly intended for the preparation phase, but these interfaces can also be used to make adjustments online during the operation phase.

5.2 Interactive and Weakness-Aware Prototype Enhancement for Fine-Grained Detection

The architecture presented in this section extends the one described in [Section 4.4](#) for fine-grained few-shot object detection. The distinctive feature of these extensions is that they add interactive mechanisms to the prototype selection procedure. This includes prototype inspection, validation, and refinement. The architecture itself is inspired by the Detect Everything Vision Transformer (DE-ViT) model [88], which works with class prototypes created by a pretrained ViT backbone. How DE-ViT can be extended to include part-level prototypes to enable fine-grained detection capabilities is described in [Section 4.4](#) as well. While there the part prototypes are selected automatically based on scores that take other similar categories into account, in this section the automatic prototype enhancing process is replaced by an iterative procedure in which the operator has full control over which parts are selected for the enhancement of the class prototypes. The operator is supported by components that visualize the model’s internal *view* of the categories and all their prototypes at hand. The goal is to keep the required expertise for using these additional tools as low as possible, so that laymen can not only use the perception system on their own but also *debug* its behavior and make adjustments if needed. For this, as in the non-interactive mode, the backbone and the detector weights are kept fully frozen, and the adjustments only operate on the collections of multi-granular prototypes that represent the categories for the task at hand. There is no fine-tuning involved at any point.

As described in [Section 4.4](#), the multi-granularity of prototypes helps to resolve ambiguity in fine-grained detection tasks. An automatic module that takes over the part prototype discovery and selection reduces the workload for the operator. However, it lacks awareness of potential context knowledge that the operator might have and that is critical for solving the task correctly. This section presents the procedure for incorporating the operator into the preparation of the DE-fine-ViT architecture described in detail in [Section 4.4](#).

Parts of this section were previously published in [P2] and build on a supervised thesis [T5].

5.2.1 Target Scenarios

In general, the target scenarios for the interactive architecture are not that different from the scenarios presented for the non-interactive architecture ([Section 4.4.1](#)): the perception model is meant for fine-grained few-shot object detection scenarios where classes can be visually very similar on a global level and only small regional characteristic features allow differentiation.

The interactive model variant is developed for scenarios where context knowledge matters but relevant image samples are missing or hard to obtain. One example use case is a logistics application in which the operator knows that in the operation environment there will be several objects of the same type, but only those with a white identification label are relevant for detection and need to be recognized. The non-interactive model would need images of these

boxes to be able to distinguish the two categories, while here the operator can just mark the sticker as an enhanced fine-grained part prototype.

Further, the automatic mode implicitly relies on certain inductive biases for choosing the most relevant part-level prototypes, which are modeled by the score metrics. If these inductive biases do not apply to the task at hand, the interactive mode provides a fallback by giving full control to the operator while remaining as user-friendly as possible to set up.

It is important to mention that if the fine-grained detection capabilities of the architecture are not relevant for a robotic application, the generic detection capabilities of DE-ViT are still fully functional. If no part prototypes are defined, the DE-fine-ViT predictions are identical to the internal DE-ViT predictions.

5.2.2 Architecture and Method

Only a few components of the architecture are adjusted or replaced compared to the model presented in [Section 4.4](#). The differences affect only the preparation phase, which is reworked to make use of the interactive aspect. Instead of the automatic class and part prototype calculations, the operator is equipped with a class-agnostic segmentation model, the Segment Anything Model (SAM) [129]. The model takes clicks or bounding boxes created by the user as input and generates object-centric masks from them. This makes it possible to quickly create pixel-level annotations of arbitrary objects with a precision sufficient for obtaining prototypes. Further, these simple user interfaces can be provided as web client applications on hand-held devices and do not require the same familiarity with the controls as other labeling software ([Section 6.2.3](#)). The prototypes, determined using the class and part masks created by the user, reflect his or her context knowledge about which characteristic features are relevant for which object category. While objects or parts might look different enough to a human, it is decisive for the final performance that the corresponding prototypes are as distinct as possible to achieve maximum robustness during inference. To make the model's perceptions, i.e., those of the pretrained backbone DINOv2, visible to the operator, simplified visualizations of the similarities among the current prototype sets are prepared on the fly. This way, the operator is aware of potential confusion between similar categories and can mitigate these risks by adding or replacing characteristic parts for those categories. In the following, the components required for this interactive preparation procedure are described in detail.

Creating Annotations

Usually, in few-shot learning tasks, it is assumed that the annotated data, i.e., the support set, already exists and only has to be processed by the model. However, in practical robotic use cases, and also in other areas, the relevant objects often appear spontaneously and can be highly specific. Publicly available datasets are therefore often not an option for obtaining the category samples. Instead, a dataset for those objects needs to be prepared shortly before the mission. To make this as easy and quick as possible, class-agnostic segmentation models

are a common feature in labeling software. By integrating not only the general segmentation model but also the detection backbone into the annotation procedure, as shown in Fig. 5.4, the goal is to reduce the effort even further. The reason is that the backbone, i.e., the pretrained DINOv2 [40], is already active and creates all prototypes as soon as the class or part masks are confirmed by the user. The set of available prototypes is then continuously assessed, and the similarity visualizations are updated accordingly. This way, the operator gains instant model insights and can adjust the annotations and the labeling strategy directly. Note that only the backbone of the architecture is used during the preparation phase, neither the detection head nor the Fine-grained Recognition Module (FRM). These network components are either kept frozen or their behavior depends explicitly on the prototypes, as in the case of the FRM.

As a class-agnostic segmentation model, any model can be used that can be controlled interactively via simple interfaces, e.g., clicks or drawn bounding boxes. It is required to be as class-agnostic as possible to provide masks even for highly specific objects. Successor models of SAM [129] have become even more performant and can replace it in the preparation phase. With SAM 3 [130], text input, e.g., obtained via speech-to-text systems, could also be used to further simplify the annotation procedure for laymen.

One of the most important things the operator has to keep in mind during labeling is that if a part is labeled on one image, it should be labeled on all other images of the same category in which it is visible. The images for a given category are grouped into sub-groups, so-called *views*, based on the available class and part labels. Therefore, if the user forgets to label a visible part that has been annotated on another image, the image at hand will be assigned to the sub-group or *view* in which this part is assumed to be not visible.

Visual Prototype Debugging

The feature that distinguishes this system from the non-interactive mode described in Section 4.3 is the interactive preparation procedure, which combines the labeling support from segmentation models with the ad-hoc assistive visualizations of the prototypes for the operator. The purpose and structure of these visualizations are the focus of this subsection.

To understand why these visualizations of prototype similarities are beneficial, it is important to understand where and which role they play inside the architecture during the inference phase (Fig. 5.4). Within the Subspace Projection module of the DE-ViT architecture [88], the dot product of the backbone features $\mathbf{F} \in \mathbb{R}^{H_p \times W_p \times D}$ with the class prototypes $\mathbf{P}_{\text{cls}} \in \mathbb{R}^{N \times D}$ and with the background prototypes $\mathbf{P}_{\text{bg}} \in \mathbb{R}^{N_{\text{bg}} \times D}$ is computed, resulting in a subspace with S_{sub} channels. These channels are post-processed to obtain a *similarity* subspace for further processing by the detection head. The subspaces are class-specific because the first channel of the subspace features is always the one corresponding to the class currently being examined by the detection head, while the other channels are locally reordered (by magnitude) and interpolated channels belonging to the other classes, and concatenated channels corresponding to the backgrounds. The subspace projection, including the reordering, is proposed by Zhang *et al.* [88] to resolve the permutation ambiguity regarding the order of classes and to

prevent overfitting on base classes during pretraining. In theory, all class-specific subspaces would need to be processed by the detection head for each proposal. To reduce the potentially large computational effort (which depends on the number of classes), an additional minimalist classifier estimates the class of each proposal based on the original ViT features $\mathbf{F}$ aligned with the Region of Interest (RoI) of the corresponding proposal, using the class prototypes as its weights. Then only the subspaces of the top- k_{det} most similar classes are forwarded to the detection head for the corresponding proposal. From the same list, top- k_{RoI} classes are also forwarded to the FRM for re-evaluating the initial prediction. If the correct category is not part of this list, it cannot be corrected later on, not even in the FRM. One extreme case is when there are more than top- k_{RoI} similar class prototypes. In this case it becomes likely that, due to subtle variances, the correct class ends up at rank top- $k + 1$ or lower and is not considered during re-evaluation. The list of top- k categories is obtained via the following procedure

$$\mathbf{f}_i = \frac{1}{H_r W_r} \sum_{H_r} \sum_{W_r} \text{RoIAlign}(\mathbf{F}, \mathbf{b}_i) \quad (5.8)$$

$$\hat{\mathbf{f}}_i = \frac{\mathbf{f}_i}{\|\mathbf{f}_i\|_2} \quad (5.9)$$

$$\mathcal{T}_i^{(k)} = \text{top-k}(\hat{\mathbf{f}}_i \mathbf{P}_{\text{cls}}^\top) \quad (5.10)$$

The part prototypes become relevant in the FRM. There they are evaluated based on their similarity with the raw cropped feature map $\mathbf{F}_{\text{bbox}}$ instead of the RoI-aligned variant. This means that if class prototypes are similar, the probability that they appear in the top- k_{RoI} list is increased, and the FRM needs discriminative part prototypes that are as distinct as possible from the parts of classes with a similar class prototype.

Based on these observations, the visualizations highlight categories with similar class prototypes. If a pair of highly similar class prototypes is found, the similarities for all of the corresponding part prototypes are visualized as well. The operator can then quickly see whether action is required to annotate additional parts. Note that class prototypes cannot be changed. Only part prototypes can be added, removed, or replaced for a class.

5.2.3 Model Preparation

For preparation, the operator needs to complete the annotation procedure iteratively until the visualizations clearly show that there are no pairs of similar class prototypes that cannot be resolved using clearly distinct part prototypes. The more classes are similar to each other, the more labor-intensive this procedure becomes, because the user then needs to find a set of part prototypes for each class that resolves all potential confusions. For N_{sim} similar classes there are $N_{\text{sim}}(N_{\text{sim}} + 1) \cdot 1/2 - N_{\text{sim}}$ possible confusions.

The frozen DE-ViT components integrated in the proposed DE-fine-ViT architecture are pre-trained on *smaller* datasets such as LVIS [30]. In certain scenarios it can be useful to retrain

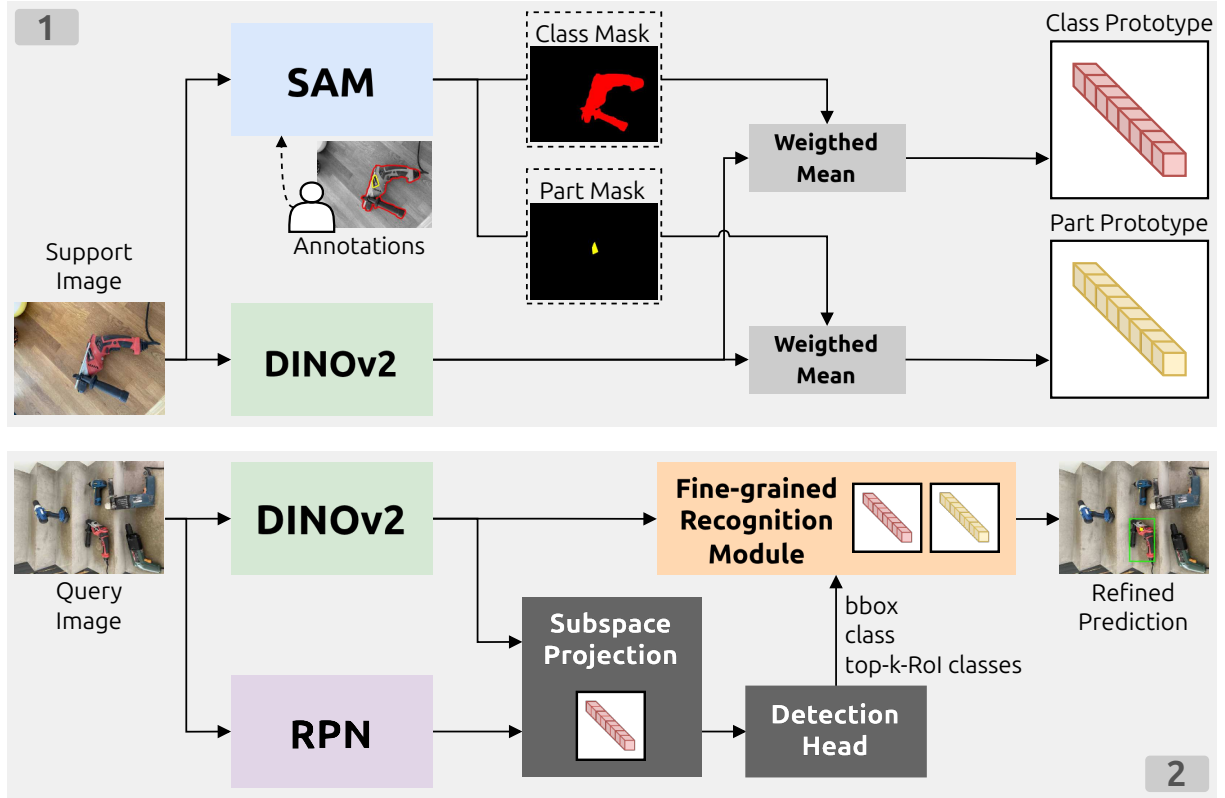


Figure 5.4: Overview of the proposed interactive Detect Everything fine-grained Vision Transformer (DE-fine-ViT) model, which builds upon the DE-ViT architecture [88] (dark gray). In comparison to the non-interactive DE-fine-ViT variant (Section 4.4), only the preparation phase ❶ is different and incorporates an operator to quickly annotate the objects and the characteristic parts of preference while being assisted by visualizations of prototype similarities (Section 5.2.2). During inference ❷, the modules and their functioning are identical to those used in the non-interactive mode.

these components on datasets that better match the planned robotic applications to better prepare the model. While pretraining the components on a specific dataset usually represents a weakness when it comes to highly specific objects in robotic applications, the detection head is designed to be feature-agnostic due to the subspace projection. It has only seen the abstracted low-rank projected subspace during pretraining, not direct image embeddings. So in theory, by creating appropriate domain-specific background and class prototypes, the detection head should not experience significant differences and might even operate on in-distribution input with respect to the projected subspace.

5.2.4 Inference

For inference, the class prototypes are used by the subspace projection block and the FRM. The latter also uses the part prototypes to refine the internal predictions even when the categories are very similar at the object level. The pipeline is then completely identical to the non-interactive inference described in Section 4.4. Interactive components and procedures are intended for the preparation phase. However, the visualizations can also be used during inference to monitor the active internal prototypes.

If false detections are observed at test time, the operator can return to the preparation phase and directly make modifications for the class or classes where the model suffers from confusions. Adjustments to the part or background prototypes can then serve as the lever for changing the system’s behavior.

Table 5.2: Comparison of the non-interactive and interactive DE-fine-ViT modes.

Aspect	Non-interactive Mode	Interactive Prototype Selection
Primary Objective	Improve fine-grained discrimination by introducing multi-granular (class and part) prototypes	<i>identical</i> , improved context awareness through human-guidance
Part Prototype Creation	Automatic construction from annotated support images (bbox labels)	Interactive annotation with segmentation model
Handling of fine-grained Ambiguity	Score-based during preparation	Interactive similarity analysis and corrective actions
Prototype Grouping	View-dependent sub-groups inferred from unique combinations	<i>identical</i>
Part Relevance Weighting	Fixed or uniform weights assumed	Optional adjustment of part weights
Model Adaptability	Adding or removing labeled data, hyper-parameters	<i>identical</i> , additional iterative prototype revision
Backbone and Detector	Frozen backbone and detector, no fine-tuning	<i>identical</i>
Computational Overhead	Minimal overhead from part-based re-evaluation	<i>identical</i>

5.3 Summary and Conclusion

The methods presented in this chapter extend the resource-efficient perception approach of this work by introducing interactive mechanisms that incorporate operator context knowledge directly into the perception process. While the previous chapter focuses on minimizing or eliminating training through the use of pretrained representations, this chapter addresses a complementary limitation: the ambiguity and underspecification that arise when only few examples are available. The central idea is therefore not to optimize model parameters or add more training data, but to improve task alignment by enabling the operator to guide the model’s decision-making process in an intuitive manner. A key contribution of these methods is the design of interactive interfaces that are tightly coupled with the underlying perception architectures. These interfaces expose intermediate representations of the model, such as attention maps, feature activations, or part-based correspondences, in a form that is more accessible to non-expert users. Instead of treating the model as a black box, the operator can inspect how decisions are formed and identify mismatches between the model’s focus and the task-relevant features.

The methods introduce mechanisms that translate operator feedback into modifications of the inference process. In the context of few-shot recognition, this is realized through the adjustment of feature importance and the conceptual refinement of prototype representations. Rather than adding new data or retraining the model, the operator can emphasize or suppress specific visual characteristics that are critical for distinguishing between categories. This reshapes the similarity relationships during inference and aligns them more closely with the semantic distinctions required by the task. For fine-grained detection scenarios, where categories may differ only in subtle local details, the interaction is extended to explicit guidance. Here, the operator can influence which regions or parts of an object should be considered most discriminative. The perception pipeline incorporates this feedback by reweighting or filtering detections based on part-level correspondences. This enables the system to focus on the relevant structures, even when these are not sufficiently represented in the limited training examples.

Both types of methods have in common that pretrained models provide rich but generic feature representations, while lightweight, interaction-driven modules adapt these representations at test time. This avoids the need for iterative training cycles and enables rapid reconfiguration of the system for new tasks or changing requirements. The interaction mechanisms are designed to be incremental and reversible, so that operators can iteratively refine the model behavior without risking degradation or erroneous updates. However, the effectiveness of the approach depends on the operator’s ability to correctly interpret model behavior and provide meaningful guidance.

In conclusion, the methods of this chapter establish interactive perception as a complement to resource-efficient model design. They shift the adaptation process from data-driven learning to human-guided refinement and enable flexible adjustments even after deployment, in scenarios where traditional training-based approaches might be impractical.

6 Evaluation

This chapter presents the evaluation of the methods introduced in [Chapters 4](#) and [5](#). Since some of the model architectures in those chapters are partially identical and differ only in their mode of operation, the experiments and their results are presented together so that mode-specific strengths and weaknesses can be compared directly alongside the method-specific evaluation. This is covered in the first section ([Section 6.1](#)). The subsequent section ([Section 6.2](#)) gives an overview of the qualitative results obtained in various research projects. Some of the proposed methods were developed specifically for the core robotic vision tasks of those projects. The main goal of this research was to develop perception systems that address different object-centric robotic vision tasks while requiring fewer resources, such as data, time, expertise, and hardware, than existing approaches. The strengths, weaknesses, and shortcomings are assessed in the final discussion ([Section 6.3](#)) with reference to the research questions stated in [Section 1.2](#). Current limitations are outlined with respect to the overall objectives of this work.

6.1 Evaluation of Proposed Architectures

This section focuses on the evaluation of the proposed methods and architectures. Common benchmarks are used where available to obtain quantitative results and to enable comparison with other state-of-the-art approaches. Some architectures have a target application, e.g., segmentation with frozen vision transformers was developed for litter segmentation. In such cases the evaluation specifically addresses the corresponding scenarios. Qualitative results for special use cases are also presented. A conclusion is drawn at the end of each section. In the discussion at the end of this chapter (Section 6.3), these conclusions are revisited in a broader context.

6.1.1 Frozen Vision Transformer for Attention-Based Segmentation

The developed architecture aims at leveraging frozen pretrained models for extremely heterogeneous categories such as litter. Creating a representative dataset for litter objects is nearly infeasible, because almost any item can belong to the waste category. Therefore, the architecture is designed to treat litter as something anomalous that does not belong to a set of known classes. This strategy avoids the need to collect images of litter. Instead the model needs to be pretrained on classes that are rather common, expected to be found on the ground, and not classified as waste items in the robotic scenario at hand. In this evaluation section, the performance of this concept is analyzed on a commonly used trash detection benchmark, and it is examined whether the inherent capabilities of a publicly available pretrained Vision Transformer (ViT) are sufficient to perform the task without any specific training on large trash datasets.

Parts of this section were previously published in [P1] and build on a supervised thesis [T8]. More implementation details can be found in Section A.1.

TACO Evaluation

As described in Section 4.1 the architecture, depicted in Fig. 4.2, is built around a pretrained ViT. The model is the ViT-B [12] variant that has 12 encoder layers, approximately 18 GFLOPS, and 90 million parameters [131]. These limited resource demands allow edge deployment, e.g., on a mobile robotic platform. The ViT was trained on the ImageNet-1k dataset [4] for the image classification task. The model processes input images resized to a resolution of 224×224 , with a patch resolution of 14×14 and 16×16 pixels per patch. With little effort this model can be replaced by another vision transformer, e.g., one with a higher patch resolution that would also increase the input image resolution. In one experiment, the same ViT variant pretrained on the larger ImageNet-21k [5] was used for comparison, but performed slightly worse. Therefore, the following experiments are all based on the ImageNet-1k pretrained ViT-B model.

Figure 6.1 visualizes the attention for each of the 12 encoder layers on an example input image from the TACO dataset [117] for trash detection. The extracted attention maps are regular

mean attention maps calculated from the corresponding head attention maps (Section 2.3.1). Many approaches use attention maps for foreground segmentation. The same capability is exploited here to segment litter on the ground. Various types of grounds were part of the pretraining dataset, and the model should tend to ignore them since, on average, they do not convey information about the correct class. In the figure, it becomes clear that the attention maps in the first layers do not focus solely on the trash item. However, layer by layer the focus becomes stronger and reaches a peak in layer 4. In contrast, for higher layers the attention maps become harder to interpret. Only in the last layers is the target item’s outline noticeable again, even if only to a small extent.

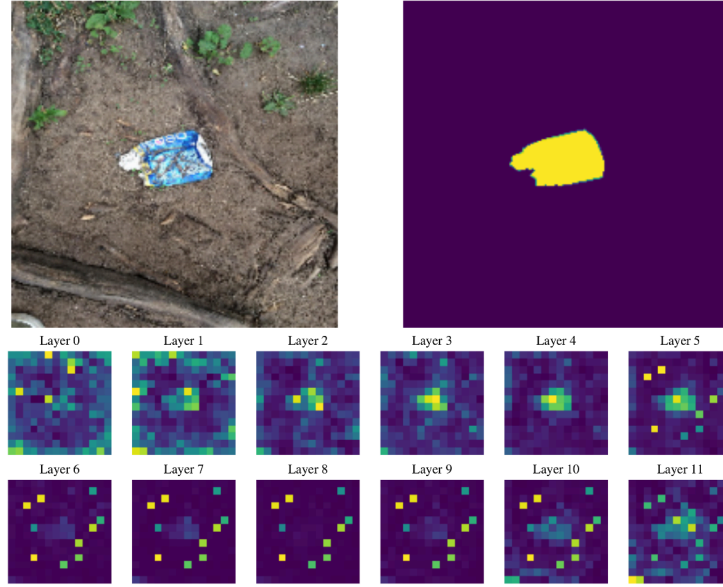


Figure 6.1: The input image from the TACO dataset and the ground-truth mask [117] at the top. The raw layer-specific mean attention maps of the vision transformer backbone on the bottom. In this and in other experiments, layers 2-5 qualitatively show the highest correlation with the trash objects. Note that the color maps are specific for each layer attention map to better visualize the different value ranges. Copyright ©2025, IEEE [P1].

Potential binary masks that can be derived from these attention maps are shown in Fig. 6.2. Note that the attention maps are not calibrated, so the value at each patch cannot be taken as a probability prediction. Instead, a threshold needs to be determined for each layer. In Fig. 6.2 these were obtained by computing the layer-specific threshold $\tau_{\text{seg}}^{(\ell)}$ according to the best F1 score $F1_{\text{max}} = \max_{\tau_{\text{seg}}^{(\ell)}} F1(\tau_{\text{seg}}^{(\ell)})$ using Eq. (2.4) and the given ground-truth mask (top right in Fig. 6.1). The mask at layer 4 provides the most accurate segmentation. To obtain these masks, the attention maps, which have the same resolution as the patches of the ViT, are upsampled to the input image size.

For a quantitative analysis, the TACO dataset [117] is used as the foundation since it provides realistic and diverse images as well as pixel-level annotations with multiple sub-categories. In addition, the dataset assigns each image to a certain scene type. For all experiments in this work, these scene type labels were used to filter the 1500 images of the dataset and keep only those of the scene types pavement, sand, trash, vegetation, and water. For evaluation the Area

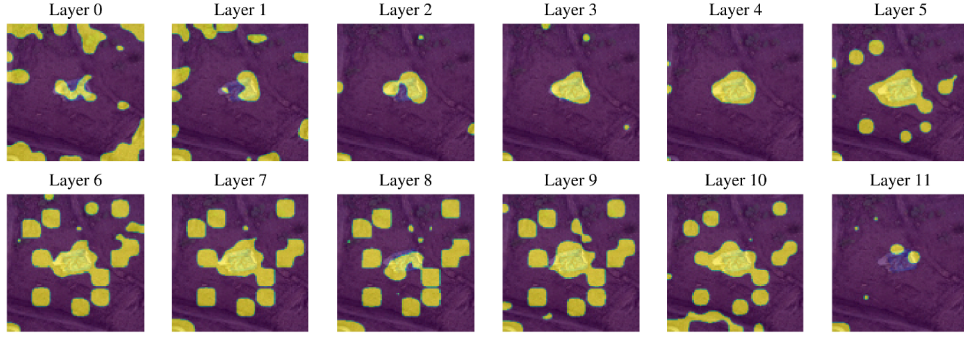


Figure 6.2: The segmentation behavior of the plain vision transformer (ViT-B-224) varies across layers. Qualitative results show that layers 2-5 focus most strongly on larger trash objects, with layer 4 producing the most accurate segmentation. Figure from [T8].

Under the Receiver Operating Characteristic curve (AUROC) score defined in Eq. (2.8) is chosen as a threshold-independent metric. An analysis in the Receiver Operating Characteristic (ROC) space is preferred over the PR space here, since *false alarms* are considered not harmful in the context of litter segmentation. The results for the True Positive Rate (TPR) and the False Positive Rate (FPR) (Eq. (2.7)) are visualized in Fig. 6.3 and confirm that layer 4 clearly has the best capabilities for the task of segmenting waste items. Note that a perfect mask prediction corresponds to an L-shaped curve rotated by 90° clockwise, whereas a random predictor produces the diagonal line from (0,0) to (1,1).

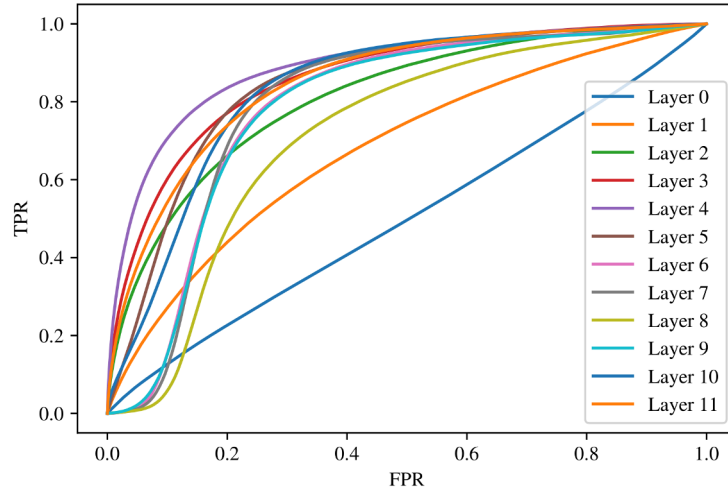


Figure 6.3: Overview of layer-specific segmentation performances on a subset of the TACO dataset [117] of samples belonging to the scene types pavement, sand, trash, vegetation, and water. Copyright ©2025, IEEE [P1].

To further optimize the attention map from layer 4 for the task of trash segmentation, a few post-processing steps are integrated, as can be seen in the architecture overview in Fig. 4.2. They are assessed individually in the following. Border cropping, where the outermost patches are set to zero, is motivated by artifacts in the form of high attention values at the edges. Adding border cropping increases the plain performance from $F1_{\max} = 0.38$ to $F1_{\max} = 0.42$ while keeping almost the same AUROC score. This is an $F1_{\max}$ improvement of approximately

4%. It is only beneficial because the object locations in the dataset are biased towards the image center. Since the AUROC score is almost identical, the improvements are mainly related to changes in the precision metric.

The module that upscales the attention maps has the sole purpose of obtaining segmentation maps with the same size as the input image. It does not affect the overall performance. However, 2D Gaussian filtering is also performed on the upscaled attention maps, to see whether the smoothing effects help the quality of the final segmentation. The results for different σ values of the filter are listed in Table 6.1. Note that the kernel size k_{GF} scales with $k_{GF} \propto 8\sigma$. Interestingly, larger values of σ appear to consistently improve the AUROC and the $F1_{\max}$ score. However, a more thorough inspection of the results showed that, while it improves the overall performance, the capability to segment smaller objects is lost, since they get *blurred out*. Nevertheless, if smaller objects are not expected in the final application scenario, a larger σ value can be considered.

Table 6.1: Pixel-level performance for different σ values.

Setting	Pixel-AUROC	$F1_{\max}$
No Filter	0.890	0.380
$\sigma = 1$	0.890	0.380
$\sigma = 2$	0.891	0.381
$\sigma = 3$	0.892	0.382
$\sigma = 4$	0.893	0.384
$\sigma = 5$	0.894	0.386
$\sigma = 6$	0.894	0.387

By default, the 12 head attention maps with respect to the class token are averaged for each layer to obtain the final fused attention map. In theory, there are various ways to determine the fused segmentation map from the heads of a layer. Here, the mean variant is compared with the min and max variants. For this, the final map is calculated according to the following equations. Starting with the head-specific attention map values $m^{(\ell,h)}(i, j)$ of head h in layer ℓ at patch location (i, j) , the three alternatives mean, max, and min over all heads of the same layer determine the final attention map value $m^{(\ell)}(i, j)$.

$$m_{\text{mean}}^{(\ell)}(i, j) = \frac{1}{N_{\text{heads}}} \sum_{h=0}^{N_{\text{heads}}-1} m^{(\ell,h)}(i, j) \quad (6.1)$$

$$m_{\text{max}}^{(\ell)}(i, j) = \max_{h \in \{0, \dots, N_{\text{heads}}-1\}} m^{(\ell,h)}(i, j) \quad (6.2)$$

$$m_{\text{min}}^{(\ell)}(i, j) = \min_{h \in \{0, \dots, N_{\text{heads}}-1\}} m^{(\ell,h)}(i, j) \quad (6.3)$$

The mean noticeably exceeds the performance of the other two, as can be seen in Table 6.2. In future work, one could specifically analyze the capabilities of each head for the segmentation

task. Alternatively, masking a subset of heads might be an option to outperform the mean fusion method, which is used as the default in all experiments in this work.

Table 6.2: Segmentation performance for different head fusion methods.

Aggregation	Pixel-AUROC	F1 _{max}
$m_{\min}^{(\ell)}$	0.813	0.290
$m_{\text{mean}}^{(\ell)}$	0.890	0.380
$m_{\max}^{(\ell)}$	0.854	0.310

Generalization Characteristics

A quite relevant aspect in the application of trash segmentation is that the variability of waste items is vast and cannot be concretely specified. One exception is when the number of sub-categories is limited to a finite, rather small number. For instance, when it is defined from the beginning that only crown caps, coffee cups, and cigarette butts are targeted. However, a more versatile cleaning robot is usually preferable. This is the motivation for the next experiments, which examine how well the model can handle *unseen* classes. Unseen is not fully accurate here, since the presented perception model is never specifically trained on trash items of the TACO dataset, only generally pretrained on the ImageNet-1k dataset. But the categories of the ImageNet dataset include items that are likely to be found as trash on the ground and thus are also part of the TACO dataset, e.g., bottle caps, carton, or plastic bags. Still, there are classes in the TACO dataset that are not part of the ImageNet-1k dataset (Fig. 6.5). Note that it cannot be guaranteed that instances of them are not present in the background of any image of ImageNet. So the ViT might have *seen* them, but it was definitely not trained to recognize them.

A conventional way to do trash recognition is to train a detection model. This is used as a baseline for the next experiment, visualized in Fig. 6.4, where a Mask R-CNN (Mask R-CNN) model is trained on the TACO dataset. First including all categories, and then in a second run, the class *Bottle* is excluded from any training images. The evaluation is then performed on a test split that contains all categories, including the images with bottles. It can be seen that while the Mask Region-based Convolutional Neural Network (R-CNN) trained with bottles outperforms the presented ViT-based segmentation model, it performed worse when bottles were not part of the training data. This shows that training works only with known classes and that there is no generalization to litter items in general. Therefore, it provides inferior performance for items unknown to the supervised trash detection model.

As mentioned before, bottles are also part of the ImageNet-1k dataset that the ViT backbone was trained on. So it practically had the chance to develop an attention-based focus towards bottles. Nevertheless, Fig. 6.5 shows qualitative results for categories that are not part of the ImageNet-1k. It becomes clear that the segmentation method is still able to provide reasonable segmentation masks, although it was never explicitly trained on such categories, e.g., straws or floss picks.

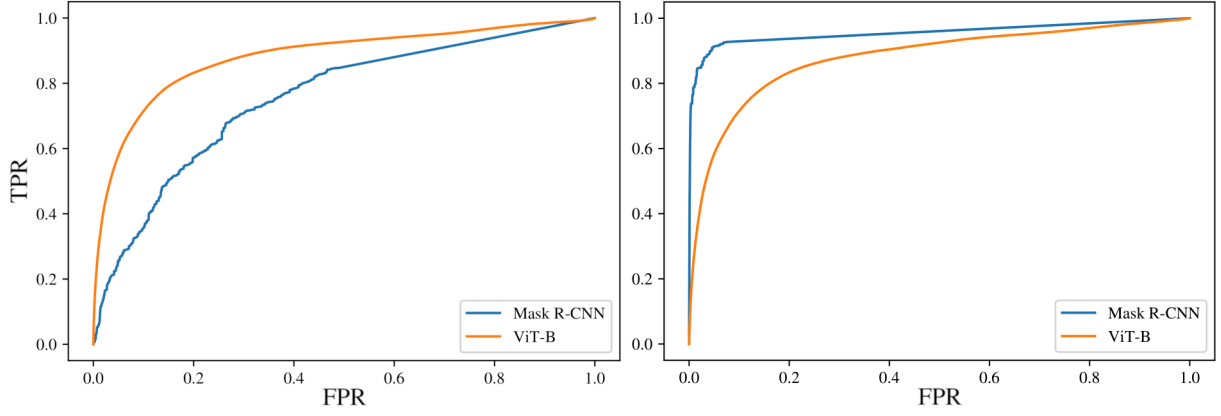


Figure 6.4: Overview of segmentation performance without the bottle in the support set (left), which is used to fine-tune the Mask R-CNN. On the right the bottle is in the support set. Both evaluations were conducted under segmentation settings on the test split containing the bottle. Note that the ViT-B based segmentation model performs identically in both scenarios since it is independent of the support set. Copyright ©2025, IEEE [P1].

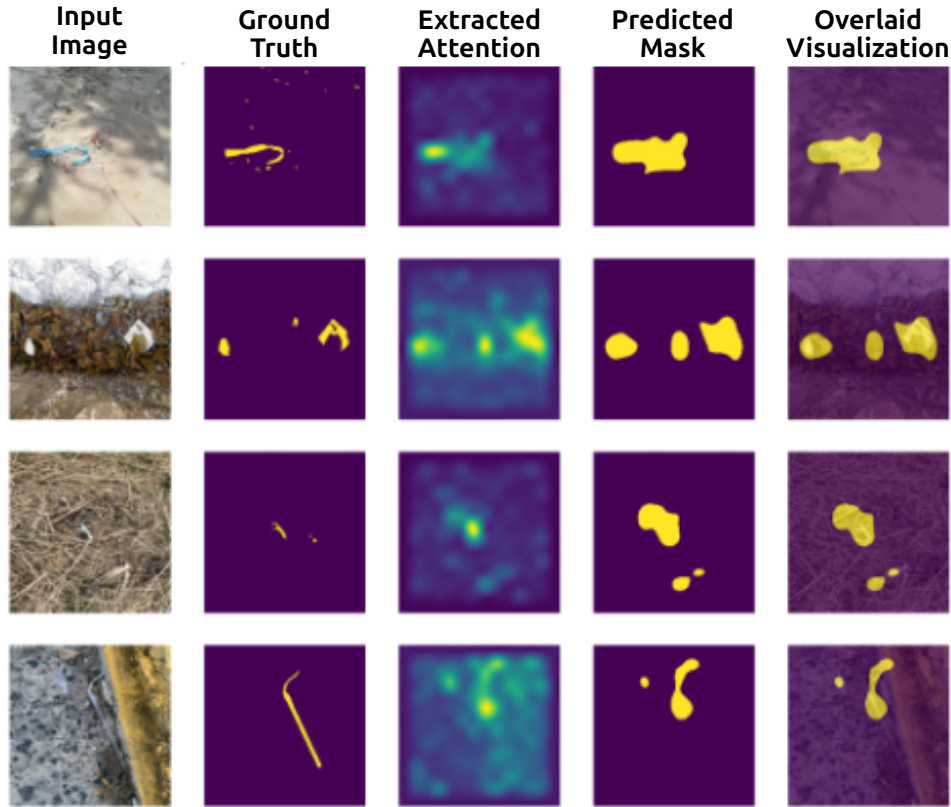


Figure 6.5: Qualitative analysis for classes that the backbone has never seen as labeled categories during any of its pretraining on ImageNet-1k. For instance, plastic nets, styrofoam, floss picks, and straws (from top to bottom). Copyright ©2025, IEEE [P1].

Distraction Branch

As a final experiment, potential use-cases for the distraction branch are examined. The previous section already described that the model was pretrained on a dataset with 1k classes.

Some of them are part of typical waste categories, but many are also regular animal species, vehicle types, humans, or even complete buildings. For this, the distraction branch is integrated. It can remove attention from the fused attention map (Fig. 4.2). Different strategies for this are evaluated qualitatively (Fig. 6.6). The most promising is class-based attention removal using the Layer-wise Relevance Propagation (LRP) method proposed by Chefer *et al.* [47]. This method analyzes the predicted class probabilities obtained from the classification head pretrained on ImageNet-1k. These probabilities do not play a role in any other component of the present architecture. However, they provide insight into the model’s perception, showing whether the model recognized one of the 1k known classes. There are two possible scenarios. For classes that belong to the litter meta-category, nothing is done. For classes that do not belong to the usual trash items, it is checked whether a certain threshold is met. If it is, the LRP method is then used to determine the class-specific attention map. This map is then subtracted from the fused attention map, resulting in an attention map with non-waste areas removed.

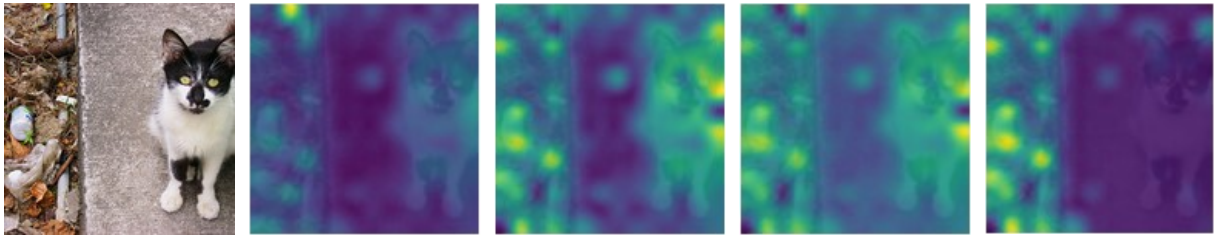


Figure 6.6: Final attention maps using different attention masking (*distraction*) strategies. From left to right: (i) input image, (ii) raw attention map, (iii) *subsequent*, (iv) *in-process*, (v) Chefer-LRP [47] masking. The right image depicts the best solution, since the attention towards the cat is redirected towards the cluttered side of the pathway. The colormap ranges differ from image to image for improved visibility. Figure adapted from [T8].

As a comparison to the LRP method, a *Max-Reduction* strategy and a strategy based on iterative masking are used. The first penalizes the $N_{MR} = 5$ highest attention value locations by a factor of 0.8, and the second completely masks out the areas belonging to those patches on the input image before processing it again. Thus the latter strategy might lead to the image being processed multiple times by the segmentation model. While these two strategies sound rather simplistic, they performed better on a TACO dataset evaluation, with the results presented in Table 6.3. When investigating the performance differences, it appeared that the distraction branch was activated too often when using the LRP method, for objects that are trash but were predicted with high probabilities corresponding to non-trash items, e.g., white bottles often seemed like various bird species to the model. With a higher image resolution this harmful behavior might be reduced, since the patches processed by the ViT would cover a smaller image area, leading to a higher perception resolution.

Conclusion

The architecture for leveraging attention maps of pretrained vision transformers for segmentation tasks was evaluated on the use case of segmenting waste items. This task has high

Table 6.3: Pixel-level segmentation performance of different attention masking (*distraction*) strategies.

Method	Pixel-AUROC	F1 _{max}
Max-Reduction	0.890	0.379
Iterative-Masking	0.854	0.352
Chefer-LRP	0.680	0.305

intra-class variability, which directly makes it hard to solve with little data. Since limited data cannot cover the extreme variability, an alternative approach is chosen by inversely defining trash as *unknown and salient* image regions. It is shown that this is advantageous compared to supervised pretrained models in scenarios where items of an unseen category appear. The precision is inferior when it comes to known categories. Still, these strengths and weaknesses would be suitable for a combined approach, where the proposed architecture serves as an extension to a trained detection model to also point out unknown objects.

The architecture’s distraction branch builds on top of the knowledge of the pretrained vision transformer to exclude regions associated with known non-waste categories, e.g., animals. However, confusions caused by high class probabilities for non-litter categories, likely due to the small image resolution, made it impossible to fulfill the branch’s purpose sufficiently. In future work, the resolution of the images could be increased to reduce false predictions and to create more precise segmentations for smaller or thinner objects. In addition, it is possible to exchange the integrated pretrained model with little effort for newer versions, pretrained on ground-related or larger datasets.

The evaluation is mostly based on the AUROC metric, which works without a threshold and comes with the risk of overestimating the performance. In practical applications defining a threshold would be necessary, which would require determining it based on the limited available data. In addition, the quantitative experiments are only conducted on one trash dataset. Although this provides a realistic initial benchmark, the results might be biased by dataset-specific properties, such as scene composition, object placement and size, or the represented litter categories. Further evaluations on additional trash datasets would therefore be required to assess the generality of the observed behavior.

6.1.2 Fine-Tuned Pretrained Models

This section evaluates the approach that combines transfer learning with a parallel depth-based processing pipeline. Parts of this section were previously published in [P5] and [P8], and reported in [P7]. Training configurations and implementation details can be found in Section A.2. Sorting batteries on a conveyor belt is chosen as the main example application because it shows the need for additional depth information during inference. However, processing depth is optional. Use cases for the approach without any depth perception, relying only on transfer learning and data augmentation strategies, are presented in Section 6.2.

There is a lack of suitable datasets that cover all relevant categories in a common recycling scenario. Therefore, a small set of domestic battery types was collected to create the basis of such a dataset. Table 6.4 describes the contained items in detail, and Fig. 6.7a shows examples, except for the large notebook lithium-ion batteries (LIBs). Due to the limited size of the dataset, the train-test splits are unbalanced and quantitatively different across all categories. One of the main priorities when creating this set was to collect batteries from many different brands for a given type. The reason is that while the geometric dimensions are mostly standardized, the visual appearance differs significantly, representing increased intra-class variability. The distinct categories AA and AAA form a pair with high inter-class similarity. When the manufacturer is the same, the two categories only differ in form factor, and mostly only textual information allows the category to be determined visually. Note that in this work, optical character recognition is not used for inference, but it would be an interesting future research direction.

Table 6.4: Collection of batteries. Adapted from: [P5], ©2023 IEEE.

Battery Category	#Models (#Brands)	Train-Test Split	Approximate Dimensions
AA	16 (12)	30%-70%	$l = 50\text{mm}$, $\varnothing 14\text{mm}$
AAA	9 (8)	44%-56%	$l = 45\text{mm}$, $\varnothing 11\text{mm}$
Coin Cell (S)	4 (4)	25%-75%	$\varnothing 20\text{mm}$
Coin Cell (L)	6 (6)	17%-83%	$\varnothing 11\text{mm}$
Camera	5 (5)	60%-40%	<i>diverse</i>
Notebook	3 (1)	33%-67%	<i>diverse</i> ($>$ camera bat.)
Sum: 43 battery models from 36 brands (16 for training, 27 for testing)			

As a second step, training data is required for the transfer learning. The architectures used for transfer learning are usually pretrained on COCO [28] or similar large-scale object detection datasets with up to thousands or more samples per class. In contrast, subsequent fine-tuning steps typically use only tens or hundreds of images. Still, creating this many images for each category would require significant time and annotation effort, which motivates the use of the

instance-based dataset augmentation strategies presented in [Section 4.2](#). This reduces the effort because only a few images per item of a category are needed and can then be artificially multiplied to reach the required scale. Depending on the expected background variability, a certain number of background template images can be used as the starting point in the augmentation process. Since the conveyor belt is a non-challenging scenario regarding distractive features that might lead to false positives, a few empty images of the conveyor belt with the most visible features are enough. On the conveyor belt used in the demonstrator, there is a section where the ends of the belt are connected together. It is almost white and shows a zig-zag pattern that is not otherwise visible. The augmentation procedure for this evaluation uses one image of the described belt connection part and another of the normal surface as template backgrounds. This conveyor surface can be seen in [Figs. 6.7a](#) and [6.7b](#).

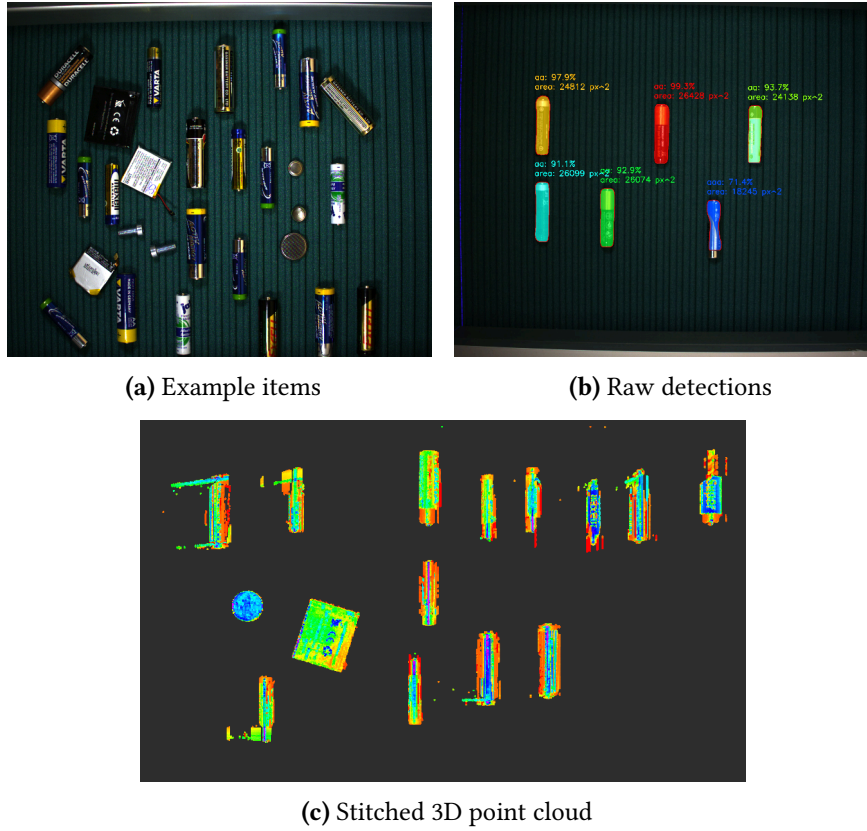


Figure 6.7: Example items (except notebook batteries), raw detections, and the stitched 3D point cloud. The raw detections represent the input to the Information Fusion Module (IFM), and will be validated with the dimensions extracted from the point cloud and the rules defined by the user. Figure (b) from [P5], ©2025 IEEE, and (c) from [P7].

Some categories may not vary in appearance as much as others, e.g., coin cells in this scenario, but they are still included to create balanced augmented training datasets. For the other categories with higher variability, the category features cannot be significantly diversified because only simple augmentation methods are applied, such as changes in brightness or contrast. While this mainly helps make the model more robust to changes in lighting conditions, the true intra-class variability cannot be replicated with this procedure when only little data is

available. Here, generative data augmentation methods that create entirely new samples can help, but they require expertise and supervision and were not in the focus of this work.

To work around the under-represented intra-class variability without falling back on generative augmentation, the model is sufficiently familiarized with the unseen categories during fine-tuning on the augmented dataset, without overfitting. The goal is to keep as much of the generalization skills as possible intact. The rationale is that the model has developed and inherently possesses these generalization capabilities, which are also transferable to new unseen categories. Instead, the boundaries of the generalization are coarsely adjusted by including negative samples as background, such as screws and pens. However, the evaluation scenario assumes that the batteries on the conveyor belt belong to one of the finite categories, so effectively avoiding false positives had less priority. In contrast, other scenarios presented in [Section 6.2](#) focus more on cluttered backgrounds with a higher risk of false positives. Regarding the generalization skills, overfitting on the few available, slightly augmented samples can be harmful and needs to be prevented. Normally, overfitting is detected when the performance on the training dataset still increases while it decreases on the validation dataset. But to detect overfitting reliably, larger training and validation datasets are required, otherwise there is a lot of noise in the measurements. Therefore, during the experiment, overfitting is avoided by choosing a small number of epochs, between 250 and 300, and a smaller learning rate. [Figure 6.7b](#) shows raw detections of the Mask R-CNN model fine-tuned on the augmented battery dataset. In addition, the upper part of [Table 6.5](#) lists quantitative results of a test run with a subset of the battery collection ([Table 6.4](#)). Both show that the model seems slightly biased toward AA, or at least often confuses AAA batteries with their larger sister type. This cannot be fixed with the false-positives-background strategy. Instead, the approach includes depth information to increase robustness.

Table 6.5: Evaluation of the perception pipeline on the test dataset. Adapted from: [P5], ©2023 IEEE.

Category	#Models	Predictions without Depth Fusion					
		AA	AAA	CC	CAM	NB	none
AA	14	93%	-	-	-	-	7%
AAA	5	80%	20%	-	-	-	-
Coin Cell (CC)	8	-	-	100%	-	-	-
Camera (CAM)	2	-	-	-	-	100%	-
Notebook (NB)	2	-	-	-	-	100%	-

Category	#Models	Predictions with Depth Fusion					
		AA	AAA	CC	CAM	NB	none
AA	14	93%	-	-	-	-	7%
AAA	5	-	100%	-	-	-	-
Coin Cell (CC)	8	-	-	100%	-	-	-
Camera (CAM)	2	-	-	-	100%	-	-
Notebook (NB)	2	-	-	-	-	100%	-

For depth-based perception, which extends the inference to a hybrid version, any depth sensor can be used. In the demonstrator for battery sorting, a line-scanner served as the source of one-dimensional scans that are stitched together based on the tracked position of the conveyor belt, as can be seen in Fig. 6.7c. Using pixel-space dimensions instead of true geometric dimensions might be robust enough and could replace costly depth sensors. However, to be robust against camera position changes, which can lead to completely different pixel-level dimensions, this is not an option. Table 6.6 describes the rules that take the geometric information into account and serve as verification inside the Information Fusion Module (IFM, Section 4.2) for the raw visual detections. With these rules active, the IFM can optimize the performance on the small practical example scenario. The performance comparison for inference with and without geometric rules is presented in Table 6.5. Creating rules before testing the operation phase of the model is possible. Due to the flexible configuration of these rules, even at runtime, they were created based on observations during the test phase in this experiment. This might not be feasible in some scenarios. For the geometric dimensions used as decision thresholds for verification, expert knowledge is needed. The operator also has to decide which geometric properties the rules work on.

Table 6.6: Set of verification rules. Adapted from: [P5], ©2023 IEEE.

Visual Prediction	Rule ID	Verification Check	Corrected Prediction
AA	R-01	$(l < l_{\min, AA})$	AAA
AA	R-02	$(l < l_{\min, AAA})$	AA
AAA	R-03	$(l \geq l_{\min, AA})$	AA
CC ^a	R-04	$(l \geq v_{\max, CC})$	Camera Battery
NB ^b	R-05	$(l < l_{\min, NB})$	Camera Battery

^acoin cell battery ^bnotebook battery

Note that in other applications (Section 6.2), using depth information was not required, so the optional depth processing branch and hybrid inference were turned off, relying fully on the augmentation and fine-tuning techniques. These applications cover tasks where the true intra-class variability is significantly smaller, so overfitting poses less of a risk than in the battery scenario. Instead, background variability and the inclusion of clutter items in the data augmentation procedure to reduce the risk of false positives were the measures used for robust model preparation. The technical details of how this is integrated into the general fine-tuning-based approach are described in Sections 4.2 and 6.2.

Conclusion

The multimodal perception pipeline shows that transfer learning can be supported by sensor fusion for objects that have typical category-specific geometric dimensions. Enhancing

a vision-based object detection model with depth sensing for verification and correction is effective for the robotic scenario of sorting batteries and helped reduce the overall number of false predictions. Although the data used for evaluation was rather small due to safety requirements for storing batteries, the results can still serve as an initial reference point for subsequent research.

The instance-based augmentation strategy is relatively easy to set up, does not require special hardware, and empirically reduced the model's overfitting on the limited training samples. However, the conditions under which overfitting is sufficiently prevented and the generalization capabilities of the fine-tuned model are not objectively determined in this evaluation, but only estimated from the observed behavior in the experiments. With a larger battery dataset and more extensive experimental repetitions, the effects of different augmentation methods could be examined in more detail and the significance of the results could be increased. Another open direction would be a comparison of the generalization behavior of different object detection models pretrained on the same dataset, e.g., COCO [28].

A known limitation is overgeneralization towards unseen categories. For instance, in the battery sorting application, pens and metal parts were used as common distractor items. One solution proposed here is to include them as background images. However, this is only an option for items that are expected. Preventing confusions with unforeseeable objects that don't belong to any known category, e.g., arbitrary waste items found among returned household batteries, remains an open challenge.

6.1.3 Part-Based Few-Shot Recognition

In this section the fine-grained object recognition architecture presented in [Section 4.3](#) is analyzed and evaluated. Parts of this section were previously published in [P3] and build on a supervised thesis [T6]. More information about implementation details can be found in [Section A.3](#). The architecture builds strongly on top of the feature representations provided by a pretrained backbone. The quality of these representations is decisive for the performance of the model’s inference. Therefore, different backbones are examined in the experiments. However, it is important to note that some components are especially optimized for ViT backbones, e.g., the foreground estimation relies on them. The multi-layer prototypes are also developed in coordination with the peculiarities of ViTs. This reliance is intentional, since ViTs have demonstrated their suitability to be pretrained on large amounts of data, even without supervision, and outperform CNN-based counterparts on down-stream tasks [36, 40]. Nonetheless, the architecture can still work with other backbones, e.g. by replacing the foreground segmentation with an additional model for that subtask, but that is outside the focus of this work.

[Figure 6.8](#) visualizes a qualitative example of a 5-way-5-shot fine-grained few-shot object recognition task with the proposed architecture. The simplest inference variant works in 4 steps: foreground estimation, part separation, part similarity, and the final scoring. In the following, the preceding preparation phase is described in detail. It starts with providing the model with 5 images for each of the 5 classes, and only image-level labels are allowed and available to the model.

During that preparation, the Support Vector Machine (SVM) integrated in the foreground estimation component was trained on the foreground estimation results of the attention rollout algorithm [46] with respect to the class token. Attention rollout ([Section 2.3.1](#)) roughly segments the foreground but is too costly for online inference. With a trained linear SVM for each of the 5 classes, the patches can then be quickly classified as class-specific foreground or background during the operation phase.

In the preparation phase, the foreground patch descriptors for a given class and its 5 support samples are clustered using K-Means to set up the part separation module. The descriptors of one cluster are all assigned to that corresponding part, which is not necessarily a defined subcomponent of the object like a handle or a battery pack, but still correlates with the structure of the object and is semantically consistent across related categories. For instance, the wings of birds are perceived as being similar to the wings of planes. The model sees them as the same part across similar classes. It is important to mention that the descriptors in this step $\mathbf{f}_{i,j} \in \mathbf{F}_O$ always consist only of the output-layer features for each patch. Other facets and other layers are specifically never used for this purpose, since the output tokens possess the best understanding of object structure. The class-agnostic structural understanding of objects by vision transformers is analyzed in depth in [38]. Empirically, it became clear that the number of clusters N_p , which represent the parts, should not be too small, to avoid missing fine-granularity, and not too large, to keep the comparison tractable. This parameter seems to have a category-specific correlation. For the example in [Fig. 6.8](#) from the CUB-200-2011



Figure 6.8: Visualization of the inference pipeline of the proposed fine-grained object recognition architecture. The shown task is a 5-way-5-shot task from the CUB-200-2011 dataset [26]. In (a) the foreground is segmented, in (b) the semantic parts are separated, in (c) the part similarity with the trained classes is estimated (note the figure only shows the similarities for the first query image input 0), and (d) presents the final class score maps. On the diagonal of (d) the birds are clearly visible, demonstrating that the similarities with the correct class are the highest and that the architecture also coarsely localizes the objects within the query image. Copyright ©2024, IEEE [P3].

dataset [26], $N_p = 8$ is used. The 8 cluster centers of foreground patch descriptors for each class are normalized and serve as the weight matrix for the part separation component. The part separation (*part association*) maps $\mathbf{M}_{\text{part}}$ for each of the part cluster centers, also noted as part prototypes, are depicted in the part separation step in Fig. 6.8b. The brighter the area, the more it shares similarity with the corresponding part prototype that was determined during the preparation phase for that class.

To be able to compare the parts during inference in the part similarity component, a SVM is trained on the multi-layer descriptors $\mathbf{f}_{i,j} \in \mathbf{F}$ of each cluster during the preparation phase, since they include low-level features such as colors and textures, which are essential for a fine-grained part comparison and separate this approach from others. The part-specific SVM learns to classify descriptors in a binary setting, predicting a score for how likely the patch belongs to that part. This also avoids running costly clustering methods during inference

in the operation phase. The score maps of all these SVMs are element-wise multiplied with the part separation maps from the previous step, obtaining $\mathbf{M}_{\text{sim}}$. This can be interpreted as answering the question whether the structural part of the foreground object also looks like the same structural part of the learned category. The part similarity box of Fig. 6.8c shows the qualitative results for the first query image `input 0`. Only the part similarity modules for `class 0` clearly show recognized parts, which is as expected for `input 0`.

Figure 6.8d shows the final score maps for all classes. These maps are accumulated and averaged part scores (Eq. (4.6)) obtained in the previous steps, which result in a final class score that is multiplied with the foreground estimation map from the first step. Each row represents the results of the learned recognition model for the corresponding class. Each column represents the score maps of all recognition models for the corresponding query image. The missing component that does the part reweighting is not active in the example, so all part similarities are averaged without increasing or decreasing the relevance of a certain part. Later in this section, an experiment is presented where the part reweighting is applied interactively.

Quantitative Results

The architecture is quantitatively evaluated on the CUB-200-2011 dataset [26], which is a common benchmark for fine-grained recognition models. It contains 200 bird species and 60 images are available for each species. To obtain a typical few-shot setup, only 5-way-5-shot tasks are considered. In total, 32 of them are randomly sampled from the dataset once and used for all CUB-related experiments. The tasks only include image-level labels, unless mentioned otherwise. This is required for classification settings. However, for some experiments mask annotations are used to examine other aspects of the processing pipeline.

In Table 6.7, three backbones are evaluated: self-Distillation with NO labels (DINO) [36], its successor DINOv2 (without registers) [40], and the frozen image encoder of Bootstrapping Language-Image Pre-training v2 (BLIP-2) [44]. For each backbone, different ViT facets, such as queries (Q), keys (K), values (V), layer tokens (T), and output tokens (O), serve as the source for the multi-layer patch descriptors, which are concatenated to form the patch feature tensor $\mathbf{F} \in \mathbb{R}^{D_F}$. Further, combinations of internal facets with the output tokens by simple concatenation are also examined, e.g., queries from all layers together with the output tokens, denoted as QO. The purpose of this experiment is to evaluate the suitability of different backbones and patch feature grids for the task at hand, which is why mask labels are used here to replace the foreground estimation step. Part reweighting is in *equal* mode, meaning that all parts have the same weight and relevance. As can be seen in the table, among all facets the layer tokens (T) perform best after the output tokens (O) for the DINO family. For BLIP-2 it is different since the layer tokens and values (V) are in the last places. Combining the output tokens with the facets leads to an improvement in almost all cases, except for TO using DINO. However, the relative performance increase is only marginal with the BLIP-2 backbone. Note that the training of the DINO backbones is conducted without supervision on images only, while the pretraining of the image encoder of BLIP-2 uses multimodal vision-language data. Further, the

BLIP-2 image encoder is a significantly larger ViT than the one of the DINOv2 model variant (Section A.3.1). This might explain why BLIP-2 is leading in almost every column.

Table 6.7: Model evaluation using different backbones and mask labels (average accuracy of 32 5-way-5-shot tasks on CUB-200-2011 [26]). The single digits are the row-wise ranks. Differences in brackets or with an arrow compare facet-specific performance with and without output tokens (O).

Backbone	Q	K	V	T	O	QO	KO	VO	TO
DINO [36]	82.9 9	84.2 8	88.0 4	89.0 1	85.6 6	84.5 (+1.6) 7 ↑ ₂	85.8 (+1.6) 5 ↑ ₃	88.5 (+0.5) 2 ↑ ₂	88.4 (-0.6) 3 ↓ ₂
DINOv2 [40]	80.6 8	79.8 9	87.4 7	97.2 4	97.6 1	97.2 (+16.6) 3 ↑ ₅	93.6 (+13.8) 6 ↑ ₃	96.6 (+9.1) 5 ↑ ₂	97.6 (+0.4) 1 ↑ ₃
BLIP-2 [44]	97.2 6	97.2 7	97.9 4	96.1 9	98.3 2	98.3 (+1.1) 1 ↑ ₅	97.4 (+0.2) 5 ↑ ₂	98.2 (+0.3) 3 ↑ ₁	96.6 (+0.6) 8 ↑ ₁

After testing different backbones, a comparison with the top-performing architecture PTMAP-SF-SOT_p [132] is conducted. Table 6.8 compares the average accuracies on the 32 5-way-5-shot tasks. For this experiment no mask labels are used, to ensure the typical few-shot image classification setting. The proposed architecture uses DINOv2 as its backbone and creates the image description matrix only from single facets. The foreground estimation falls back on the attention rollout method adapted from [107] to obtain the estimated foreground maps. It becomes noticeable that this directly harms the performance, except when using keys (K) for the multi-layer image description matrix. While the drop is only slight for DINOv2-based model variants, it is significant for BLIP-2-K. When compared directly with DINOv2-K, BLIP-2-K is still marginally better, although comparing facets between models has to be interpreted with care, since the roles of the facets seem to depend on various factors, for instance on the training procedure. In Table 6.7 it is clear that the ranking of the single facets is different for each model. Nevertheless, in the following experiments DINOv2 is the main backbone, since it offers a more efficient image feature extraction and no multimodal overhead.

Table 6.8: Average accuracy on 32 example tasks based on the CUB-200-2011 dataset [26]. Here no mask labels were used, instead the proposed architecture estimates the foreground object using attention rollout [107]. The resulting relative performance difference regarding Table 6.7 (mask annotations instead of image-level labels) is in brackets. Adapted from: [P3], ©2024 IEEE.

Model	5-way 5-shot	
PTMAP-SF-SOT [132]	97.12	
PTMAP	93.99	
DINOv2-Q (this work)	79.47	(-1.13)
DINOv2-K (this work)	80.03	(+0.25)
DINOv2-V (this work)	84.92	(-2.5)
DINOv2-T (this work)	96.93	(-0.25)
DINOv2-O (this work)	97.37	(-0.19)
BLIP-2-K (this work)	82.85	(-14.32)

Visual Concepts

The proposed architecture for fine-grained few-shot object recognition also provides an interactive mode, where the operator can inspect the prototypes and even adjust them. In the following, the interpretability methods that are used to provide this functionality are examined and validated. The core of these methods is the idea of concept banks adapted from [51], based on concept activation vectors (CAVs) proposed by Kim *et al.* [50]. Concept Activation Vectors (CAVs) aim at interpreting feature vectors in a human-friendly way. In other words, they try to link user-defined visual concepts, such as *yellow* or *dotted*, with direction vectors in the model’s embedding space. For this, subsets of the Broden dataset [128] serve as the source of visual concepts. The dataset contains images representing semantic concepts annotated with image-level labels. For this work the color and texture subsets are used. In [51] they use a generative text-to-image model to obtain the desired concept-image pairs for arbitrary sets of user-defined concepts, which is an alternative way that is not limited to the finite number of concepts covered by a fixed dataset such as Broden.

For creating concept banks, one first needs to select the concepts that are relevant for the task at hand. Together they form a so-called concept set. The backbone model then processes the corresponding images assigned to the concepts. The underlying idea is that the extracted feature vectors, also denoted as descriptors, represent the concepts in the model’s language. Based on this data, CAVs can be obtained by training linear SVMs for each concept that are able to split the descriptors into the ones that belong to the concept and those that do not. For training, only descriptors of the current concept space are taken into account. The parameters of a concept-specific SVM then represent the decision boundary for that concept, and the vector orthogonal to that boundary is defined as the CAV.

To validate the method and the obtained CAVs for the different backbones, two concept spaces are considered: one with 8 colors and another with 8 textures. Training and validation are conducted on subsets of the Broden dataset. For complementary experiments the small clothing dataset [133] is chosen, with the motivation to have more meaningful validation results when using a different dataset. The classification accuracies of the linear concept predictors of the color concept space on the Broden validation split are provided in Table 6.9, and the results for the texture concept predictors are enclosed in Table A.5. The accuracies in each column are averaged over 4 runs on different sampled training and validation splits. More details can be found in Chapter A. For completeness, a DINO-trained ResNet50 with 23M parameters provided by [36] is also included. It performs worse, but it is also the model with the smallest number of parameters. Both DINO variants use the ViT-B model, which has around 85M parameters [36]. By far the most parameters are found in the BLIP-2 image encoder with approximately 1B, using the default EVA-CLIP ViT-G architecture.

Supporting the idea that the proposed multilayer prototypes are beneficial for interpretable approaches that operate on low-level features such as colors and textures, facet O always performs worst across all backbones. Additionally, it can be observed that facets Q, K, and V, often completely disregarded in other approaches, are usually better than facet T for the

Table 6.9: Prediction performance using the color concept bank across different facets and backbones.

Setting		Colors								Colors		Total
Backbone	Facet	white	black	yellow	green	red	blue	pink	orange	Mean	Rank	
DINOres	t	68.84 ± 0.61	70.98 ± 0.99	70.84 ± 1.68	66.07 ± 2.33	70.34 ± 3.03	66.87 ± 1.88	60.20 ± 2.21	61.65 ± 1.30	66.97	16	
	o	51.39 ± 0.32	46.78 ± 1.49	48.31 ± 5.25	60.99 ± 2.81	48.21 ± 2.02	54.38 ± 4.45	57.03 ± 3.47	50.80 ± 1.85	52.24	21	
	q	93.01 ± 0.40	92.27 ± 2.86	88.56 ± 3.14	91.26 ± 0.09	92.11 ± 2.38	92.16 ± 0.95	77.59 ± 2.48	83.66 ± 4.36	88.83	5	
	k	92.23 ± 0.45	91.52 ± 1.91	89.10 ± 2.49	92.02 ± 0.57	91.01 ± 1.48	92.23 ± 0.80	77.66 ± 2.93	82.29 ± 4.89	88.51	7	
DINO	v	93.10 ± 2.59	92.96 ± 2.00	87.86 ± 1.94	91.68 ± 0.34	90.56 ± 1.11	90.61 ± 0.59	77.54 ± 2.27	85.79 ± 2.11	88.76	6	
	t	93.11 ± 1.55	92.54 ± 2.15	85.59 ± 2.75	88.96 ± 0.24	90.53 ± 1.56	89.24 ± 1.70	75.09 ± 1.33	83.92 ± 2.78	87.37	9	
	o	89.22 ± 2.21	89.23 ± 0.43	79.60 ± 4.69	72.28 ± 4.15	84.69 ± 0.54	81.79 ± 0.34	72.97 ± 0.58	77.61 ± 3.21	80.92	14	
	q	93.49 ± 0.28	88.69 ± 0.60	85.67 ± 4.37	88.86 ± 0.33	89.22 ± 2.20	90.92 ± 1.38	76.25 ± 0.65	80.61 ± 2.84	86.71	11	
DINOv2	k	94.97 ± 0.49	92.60 ± 0.95	89.40 ± 2.75	89.90 ± 0.58	89.46 ± 1.30	94.24 ± 1.72	76.88 ± 1.61	83.30 ± 2.06	88.84	3	
	v	91.88 ± 0.51	90.50 ± 1.23	85.92 ± 2.84	89.38 ± 0.28	88.91 ± 1.73	91.05 ± 0.94	76.28 ± 0.31	82.53 ± 3.64	87.06	10	
	t	71.70 ± 1.17	70.58 ± 1.20	69.18 ± 2.93	62.74 ± 6.68	64.85 ± 3.32	62.47 ± 2.87	58.24 ± 0.28	65.46 ± 0.90	65.65	17	
	o	70.88 ± 0.83	69.04 ± 0.40	66.38 ± 0.57	55.49 ± 7.09	66.32 ± 3.60	61.26 ± 1.79	56.45 ± 0.60	60.70 ± 3.03	63.32	19	
	qo	85.80 ± 2.12	80.53 ± 0.33	81.42 ± 0.47	67.80 ± 3.23	79.61 ± 2.25	75.46 ± 2.62	66.27 ± 1.14	72.66 ± 0.57	76.19	15	
	ko	92.79 ± 0.87	92.22 ± 1.95	88.38 ± 2.78	89.14 ± 1.47	90.63 ± 1.08	91.13 ± 2.18	79.92 ± 0.31	82.75 ± 0.67	88.37	8	
	vo	89.60 ± 0.44	89.04 ± 0.13	85.25 ± 1.38	83.43 ± 3.96	86.92 ± 1.80	83.59 ± 2.63	73.35 ± 2.12	78.10 ± 0.71	83.66	13	
	to	72.72 ± 1.22	69.72 ± 0.58	68.26 ± 1.50	60.80 ± 5.63	65.70 ± 3.22	62.10 ± 2.18	57.90 ± 1.06	64.55 ± 1.37	65.22	18	
	q	91.78 ± 0.92	94.13 ± 0.59	92.53 ± 1.47	93.10 ± 1.16	91.29 ± 1.99	91.55 ± 1.31	78.42 ± 0.14	89.18 ± 4.60	90.25	1	
	k	90.46 ± 1.98	93.31 ± 0.96	91.36 ± 0.56	92.51 ± 1.96	91.52 ± 1.74	89.69 ± 0.88	76.89 ± 1.35	84.90 ± 2.27	88.83	4	
BLIP-2	v	87.02 ± 1.28	89.56 ± 1.44	87.21 ± 0.54	91.06 ± 0.89	87.54 ± 2.06	83.12 ± 0.98	71.58 ± 2.43	81.61 ± 2.14	84.84	12	
	t	91.62 ± 1.80	94.45 ± 0.80	90.87 ± 1.90	94.28 ± 0.82	90.60 ± 2.19	90.66 ± 0.62	74.69 ± 0.98	88.46 ± 1.04	89.45	2	
	o	55.74 ± 1.17	67.06 ± 3.72	59.92 ± 1.97	63.30 ± 5.08	60.37 ± 4.12	57.36 ± 0.64	52.38 ± 2.56	59.25 ± 0.86	59.42	20	

DINO ViTs. Interestingly, for the BLIP-2 vision encoder, facet T is almost on par with the best-performing Q, while leaving the other facets K, V, and O behind. For the color concept space predictors, it is also examined whether combinations of multi-layer prototypes with the output tokens O are superior, but it can be clearly seen that for this task combining them leads to a decrease in performance. Further, black, white, and RGB colors seem to be the *easiest* prediction tasks across all backbone candidates, and pink the most difficult one. A confusion matrix would be an interesting next step for a follow-up analysis on that observation. All in all, there is no clear winner in the color concept space: the top 8 predictors are within a range of less than 2 points. Similar observations are made for the texture concept space and can be looked up in [Section A.3.3](#).

To qualitatively validate the color CAVs, the small clothing dataset [133] images are sorted based on the different concept activations. The concept-specific activation is calculated using the mean of the top 20% patch-wise predictions for each image. [Figure 6.9](#) shows the first three and the last three images after sorting for three different color and texture concepts. Note that each concept predictor is trained to predict the presence of a concept using all other concepts as negative samples. So while the first three images seem to clearly reflect the concept, there is no shared concept observable for the last three, except for the concepts black and striped, which might be coincidence.



Figure 6.9: Sorting of the small clothing dataset [133] based on the mean of the top 20% values for each concept-specific predictor. Green borders highlight the first three images and red the last three images. Figure adapted from [T6].

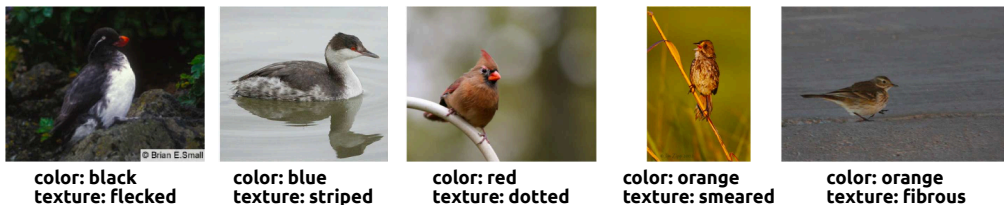


Figure 6.10: Highest concept associations for each image within the color-based and the texture-based concept space. Figure adapted from [P3], images part of the CUB-200-2011 dataset [26].

Before conducting quantitative analysis of the concept vectors, Fig. 6.10 presents an example task of the CUB-200-2011 dataset with the strongest concepts of the color and the texture concept space across all part estimators of each class. It is noticeable that these concepts are not necessarily the ones that first come to mind when looking at the images. Another explanation for some of the strongest concepts is that, since these part estimators were trained to distinguish those classes, they also focused on concepts for each class that are most unique among this set of five. At least for some of the listed concepts this explanation seems valid.

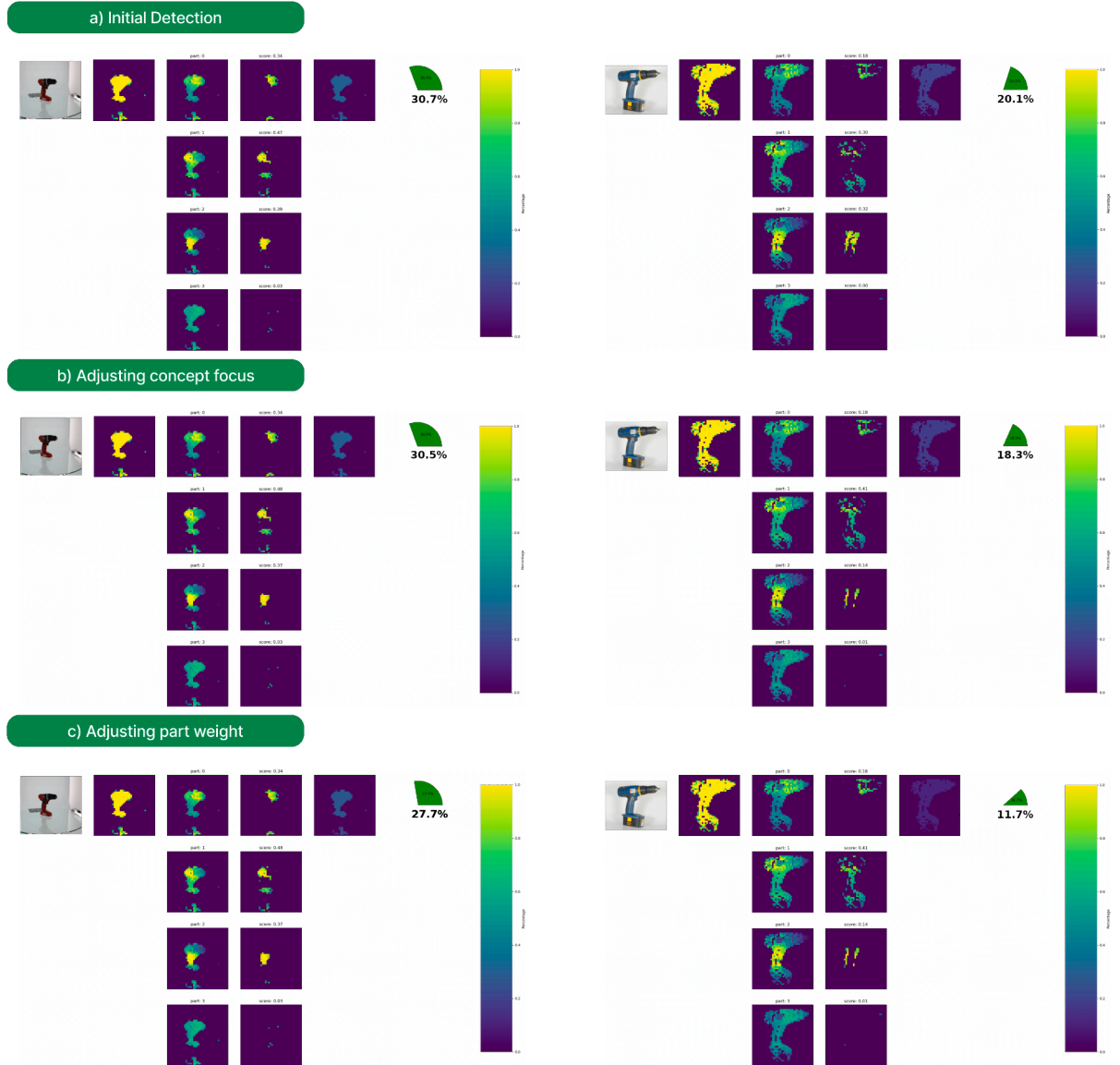


Figure 6.11: Qualitative results for an interactive teaching use case where the left power drill is to be recognized, while the right one is a false positive, which should be assigned with as little confidence as possible. Left power drill from [134]. Right power drill from [135]. Copyright ©2024, IEEE [P3].

Last but not least, a possible use case is presented in Fig. 6.11, where the CAVs are leveraged in interactive interfaces and the operator can interact with the model to adjust its focus. The model usually processes the support images and produces deterministic part estimators for a

given task. Therefore, if the input data during the preparation phase is the same, the model with its fixed inductive biases will always come to the same conclusions. The goal of the interactive interfaces is to change that by exploiting CAVs to give the operator the possibility to alter these conclusions in an interpretable way. This way the same data can still lead to different model behavior, even though the input data is identical. In the presented example, the model was trained on a certain power drill model. Subsequently, in Fig. 6.11 two query images are presented to the proposed model, one from the correct class and one from another power drill type as a negative example. One option would be to obtain a few images for this power drill as well and include it as an additional class in the support set. However, for the qualitative demonstration of the capabilities of the interactive interfaces, this is not considered. Instead, the CAVs are used to modify the part similarity estimators to focus more on suitable concepts from the color concept space that differentiate the two similar power drill types. In a first step (b), the focus of all parts is increased for the concept orange and decreased for the concept blue. This has a significant influence on the estimated part similarity and thus the final score. In a second step (c), the part that aligns the most with the tip of the power drill is completely ignored during inference, by setting its weight to 0 in the part-reweighting module of the proposed architecture. Additionally, the weight of part 1, which mostly refers to the upper body of the power drill, is doubled to fill the gap. This step further widens the gap between the *true* query image and the *false* query image, making the model more robust at recognizing the orange power drill in a scenario where the blue power drill might appear as well.

Conclusion

The proposed architecture combines common strategies for fine-grained few-shot recognition with special capabilities of recent vision transformer backbones. While choosing prototypical learning and part-based recognition is not uncommon for such tasks, the class-agnostic structural understanding of objects accessible by applying simple techniques on features from pretrained ViTs enables new options for these types of approaches. Part-based labels are tedious to acquire, but with the mentioned inherent part-segmentation capabilities this step is not necessary. Another rather unique approach are the multi-layer description tensors and prototypes, used for part similarity estimation.

The proposed model shows competitive performance on the common fine-grained recognition benchmark CUB-200-2011 [26]. The results indicate that the architecture can exploit subtle part-level differences between visually similar classes in a standard fine-grained few-shot setting. However, the quantitative evaluation is limited by the relatively small number of CUB-200-2011 tasks. The 32 sampled tasks give an initial indication of the achievable performance, but evaluations with substantially more tasks would lead to more reliable average performance values and a stronger comparison with related work. Furthermore, the experiments on CUB-200-2011 are based on bird species and are therefore only indirectly representative of robotic perception scenarios. More quantitative experiments on fine-grained robotics-related datasets are required to assess how well the approach transfers to the intended application domain.

In addition, the prototypes are examined for their interpretability, with the underlying motivation to provide an optional interactive mode. This mode gives the operator the possibility to assess the prototypes from a semantic perspective. A concept bank serves as the *dictionary* for translating the part prototypes into the relevance of different visual concepts. This selection of concepts needs to be chosen by the operator. Within this space, the interactive interfaces allow modifying the determined relevance for each of the parts. Based on the operator's context knowledge, the generalization behavior can be adjusted in a way that the focus of the model better reflects the appearance of the intended target category. The empirical results on the small clothing dataset indicate that the learned CAVs reflect semantic visual properties. Still, this only verifies the semantic alignment empirically, while Kim *et al.* [50] suggest additional statistical verification steps for CAVs, which are not applied here and should be considered in future work.

Since the concept space only has a limited number of concepts, the focus can only be adjusted relative to the concepts that are part of the space. This way, by increasing the relevance of one concept, the relevance of other concepts is always influenced as well. While these changes are measurable for concepts within the concept bank, they remain hidden for concepts that are not part of it. Nevertheless, it provides a way to actively adjust and correct how the model perceives and represents each category. The value of this is shown in an example use case, where the recognition of a fine-grained category is made more robust by improving the alignment of the part prototypes with its characteristic visual appearance. The interaction capabilities are only examined qualitatively on selected scenarios, which underestimates the complexity of realistic interaction processes and their various failure modes. To use these interfaces effectively, suitable negative samples are important, which might not always be available. Another limitation concerns the task setting itself: the current parallel class inference does not produce calibrated prediction scores, which makes open-set recognition and the rejection of unknown categories problematic. This limitation is not unique to the proposed architecture, since many other few-shot image classification methods also assume a closed-set setting.

6.1.4 Fine-Grained Few-Shot Detection

The previous evaluation section focused on the architecture for few-shot object recognition with and without optional operator interaction. From a high-level point of view, this section covers a similar concept but the task is object detection. This is a bit more challenging since in this task the objects typically cover only a small area of the image. Additionally, they not only need to be recognized but also localized. Compared to object recognition where there is one object per image, in object detection many objects of multiple categories can be contained in the same image.

The architecture subject to the following evaluation is presented in [Section 4.4](#). Parts of this section were previously published in [\[P2\]](#) and build on supervised theses [\[T4, T5\]](#). The interactive interfaces that are designed to enhance the model’s capabilities more towards fine-grained detection are explained in [Section 5.2](#). In the following, first, qualitative examinations of the state-of-the-art Detect Everything Vision Transformer (DE-ViT) model [\[88\]](#) for few-shot object detection without fine-tuning build the foundation. Second, the proposed approach is evaluated on newly created small example datasets that take a deeper look at scenarios where the inter-class similarity is high. In the same context the usefulness of the interactive interfaces is assessed. Since they are optional and work on an additional level of prototype granularity, they are not meant for generic detection benchmarks where the categories are rather coarse with moderate inter-class similarity.

DE-ViT Weaknesses

The recently published DE-ViT model [\[88\]](#) ([Section 3.1.3](#)) for few-shot object detection only requires a few samples. For this it leverages the DINOv2 vision transformer [\[40\]](#) as its feature extractor and creates class prototypes based on them. This ViT backbone has been trained without supervision and is able to perform competitively on image classification via a simple k-NN algorithm on ImageNet-1k [\[40\]](#), which means that the embeddings inherently come with a good visual understanding of diverse categories. DE-ViT compresses each category into a single embedding vector which the detection head uses for class recognition and region refinement. The benefits are simplicity and good generalization even from a few samples. While being useful for generic categories, this is a disadvantage in fine-grained scenarios when the prototypes for two categories end up being very similar, leading to false detections ([Fig. 6.12](#)). Objects that look similar but belong to a different application-specific category cannot reliably be differentiated. The proposed architecture ([Section 4.4](#)) tries to mitigate this weakness by including additional prototypes that store part-level features. Possible part-level characteristics for the 4 categories that represent the created mini-tool-dataset are depicted in [Fig. 6.13](#). The number of additional characteristics used for part-level prototypes is arbitrary and can also be zero. If no category has any part prototypes, the proposed architecture works exactly like the original DE-ViT model.

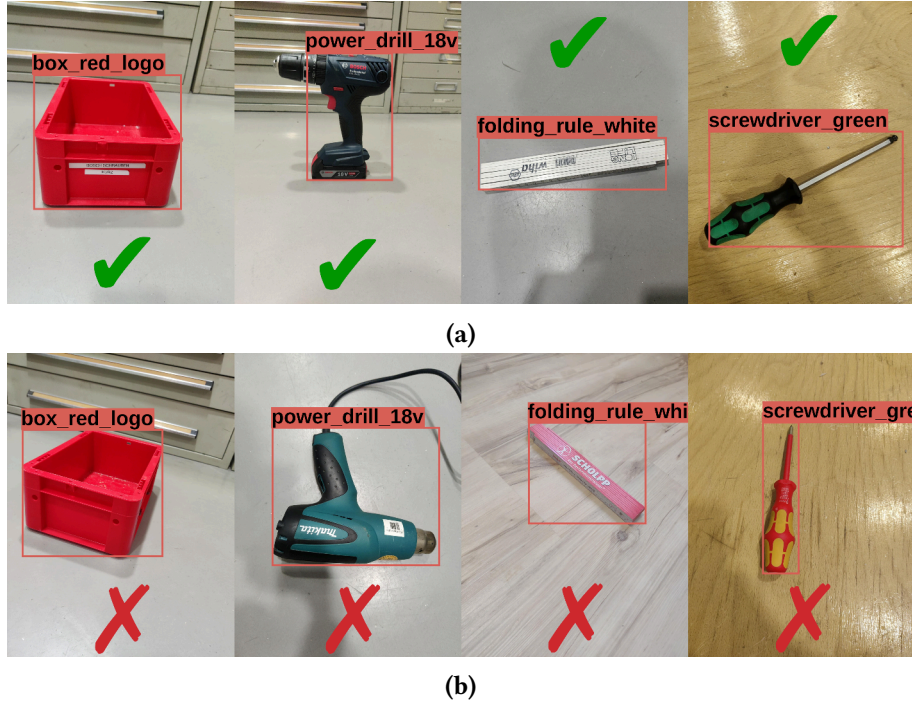


Figure 6.12: DE-ViT weaknesses regarding similar object categories. The bottom row are false positives due to the items having a high class-level prototype similarity with the corresponding ones in the top row. Copyright ©2025, IEEE [P2].

Quantitative Results

First, the proposed Detect Everything fine-grained Vision Transformer (DE-fine-ViT) model is compared with the original DE-ViT model [88] on the mini-tool-dataset, specifically created to initially examine the extended model capabilities. It consists of 4 different objects. Among them are two pairs of similar objects. One pair is the same type of boxes except for one having a small white label on its front. The other pair consists of two similarly structured hand-held tools with backbone embeddings that have a high cosine similarity. The dataset comprises a support set with 40 and a test set with 152 images with 253 mask annotations captured in a workshop environment with little to no clutter. Further, the objects are always depicted from the same side (± 60 degrees). For the support dataset all category-specific parts are annotated on a pixel level. Note that the DE-ViT model does not process any part-level annotations.

In Table 6.10 the mean Average Precision (mAP) values are listed for both models, and in Fig. 6.14 the part recognition for one of the categories is visualized. The parameter $\text{top-}k_{\text{det}} = 1$ is identical for all variants, while $\text{top-}k_{\text{RoI}}$ is altered for the proposed DE-fine-ViT model. This parameter only exists in the DE-fine-ViT model and controls the number of the most similar categories estimated for a given region proposal that are forwarded and processed by the Fine-grained Recognition Module (FRM). The module, described in Section 4.4, refines the DE-ViT predictions based on the corresponding part-level prototypes. Combined with the True Positives (TPs), False Positives (FPs), and False Negatives (FNs) metrics from Table 6.11 it becomes clear that if $\text{top-}k_{\text{RoI}} = 1$ the mAP is slightly increased compared to DE-ViT due



Figure 6.13: The mini-tool-dataset used for initial experiments contains two similar pairs of categories. The highlighted parts are used for evaluation of the interactive mode of DE-fine-ViT. Copyright ©2025, IEEE [P2].

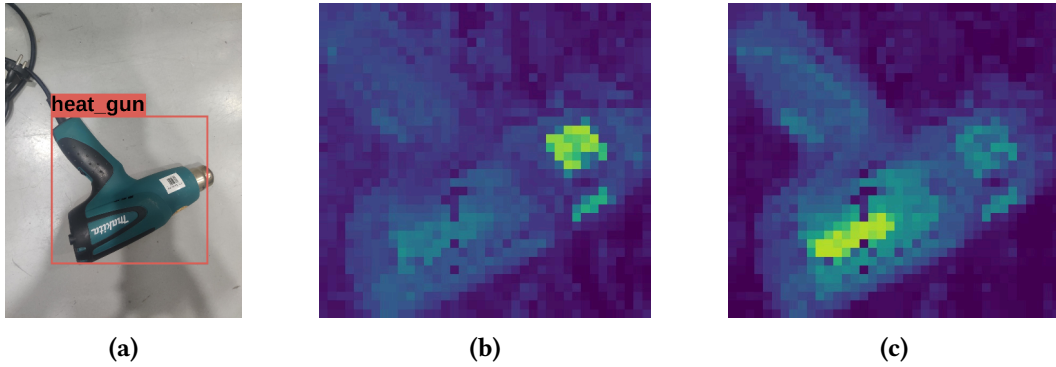


Figure 6.14: Visualization of the part-based refinement leading to the final prediction (a) based on the parts logo (b) and barcode (c), highlighted in Fig. 6.13, recognized within the bounding box. Figure adapted from [T5].

to the reduction of FP, whereas TP and FN remain identical. This makes sense, since the FRM can only falsify the one category that it processes based on unrecognized parts. By setting $\text{top-}k_{\text{RoI}} = 3$ the mAP makes a larger jump. This allows the FRM to compare the top 3 most similar categories for each region proposal, which lets it switch the final prediction to any of them based on its part-based fine-grained similarity estimations. This not only reduces FP even more, but also has a significant beneficial impact on TP and FN, boosting not only the overall precision to 0.865 ($\uparrow$ 0.194) but also the recall to 0.909 ($\uparrow$ 0.166) compared to the DE-ViT model.

Table 6.10: Results on the mini-tool-dataset with 2x2 classes. Adapted from: [P2], ©2025 IEEE.

Model (for all $\text{top-}k_{\text{det}} = 1$)	mAP@50:95	mAP@50	mAP@75
DE-ViT [88]	0.463	0.604	0.497
DE-fine-ViT $\text{top-}k_{\text{RoI}} = 1$ [P2]	0.534	0.688	0.568
DE-fine-ViT $\text{top-}k_{\text{RoI}} = 3$ [P2]	0.700	0.870	0.744

Motivated by these promising results, another dataset is created which completely focuses on a fine-grained detection scenario using 5 different power drills in more challenging and realistic environments. The dataset has 30 support images per category and a test set containing 15 images with multiple objects in different workshop and outdoor environments with backgrounds such as paving blocks, wooden plates, or concrete. There is little clutter on the test

Table 6.11: Predictions on the mini-tool-dataset with 2x2 classes. Adapted from: [P2], ©2025 IEEE.

Model (for all $\text{top-}k_{\text{det}} = 1$)	TP↑	FP↓	FN↓
DE-ViT [88]	188	92	65
DE-fine-ViT $\text{top-}k_{\text{RoI}} = 1$ [P2]	188	49	65
DE-fine-ViT $\text{top-}k_{\text{RoI}} = 3$ [P2]	230	36	23

images but at least 3 objects of different categories per image. Some of the support images include clutter such as tools (e.g., screwdrivers or brooms) and could extend the test set if the number of shots used for evaluation is less than 30. For each category between 2 and 3 part-level characteristics are labeled. All of them are listed in Table 6.12 and depicted in Fig. 6.15.

Table 6.12: Classes and parts of the fine-grained power drill dataset. Adapted from: [P2], ©2025 IEEE.

Class	Parts		
Drill 1	Chuck,	Logo,	Power Button
Drill 2	Depth Gauge,	Handle	
Drill 3	Handle,	Logo,	Switches
Drill 4	Chuck,	Logo,	Power Button
Drill 5	Chuck,	Middle Part	

**Figure 6.15:** The categories of the fine-grained power drill dataset (Section A.4.3). The highlighted parts are used for evaluation of the interactive mode of DE-fine-ViT. Copyright ©2025, IEEE [P2].

On this dataset the comparison between DE-ViT and the proposed DE-fine-ViT model is continued, with all mAP results presented in Table 6.13 and the precision and recall metrics in Table 6.14. For a deeper understanding, different numbers of class-level support images and part-level images are tested, as well as the comparison between one or two perspectives, i.e. images, for each part-level characteristic. The different model preparation settings are given in the following list:

- M1_{c10}: DE-ViT, $\text{top-}k_{\text{det}} = 1$, 10 support images per class
- M2_{c10,p10-1}: DE-fine-ViT (this work), $\text{top-}k_{\text{det}} = 1$, $\text{top-}k_{\text{RoI}} = 3$, 10 support images per class, 10 support images per class for part prototypes (1 perspective)
- M3_{c30,p10-1}: DE-fine-ViT (this work), $\text{top-}k_{\text{det}} = 1$, $\text{top-}k_{\text{RoI}} = 3$, 30 support images per class, 10 support images per class for part prototypes (1 perspective)
- M4_{c30,p10-2}: DE-fine-ViT (this work), $\text{top-}k_{\text{det}} = 1$, $\text{top-}k_{\text{RoI}} = 3$, 30 support images per class and 10 support images per class for part prototypes (2 perspectives)

- $M5_{c10,FT}$: RISF [136], 10 support images per class (+ fine-tuning on support images)
- $M6_{c30,FT}$: RISF [136], 30 support images per class (+ fine-tuning on support images)

With RISF [136] a few-shot object detection approach is included that is based on a fine-tuning procedure using the support set. This shows that fine-tuning can offer superior performance, but it comes with the cost of a more complex preparation phase, especially regarding time and hardware requirements.

Table 6.13: Detection results on the fine-grained power drill dataset. Adapted from: [P2], ©2025 IEEE.

Prepared Model	mAP@50:95	mAP@50	mAP@75
$M1_{c10}$ (DE-ViT [88])	0.246	0.345	0.304
$M2_{c10,p10-1}$ ([P2])	0.305	0.410	0.336
$M3_{c30,p10-1}$ ([P2])	0.352	0.505	0.398
$M4_{c30,p10-2}$ ([P2])	0.397	0.586	0.465
$M5_{c10,FT}$ (RISF [136])	0.674	0.883	0.772
$M6_{c30,FT}$ (RISF [136])	0.731	0.913	0.845

Table 6.14: Precision & recall on the fine-grained power drill dataset. Adapted from: [P2], ©2025 IEEE.

Prepared Model	Precision@50	Recall@50	mAR@50:95
$M1_{c10}$ (DE-ViT [88])	0.465	0.541	0.383
$M2_{c10,p10-1}$ ([P2])	0.569	0.541	0.429
$M3_{c30,p10-1}$ ([P2])	0.627	0.607	0.475
$M4_{c30,p10-2}$ ([P2])	0.632	0.705	0.540

Table 6.13 shows that variant M2 of the proposed architecture represents an improvement compared to the DE-ViT model. When working with a limited number of support images per category with the DE-ViT model, the only parameters that can be adjusted to gain performance are model hyperparameters such as $\text{top-}k_{\text{det}}$, which, on the other hand, has a high influence on computational demand [88] and can potentially hit the limits of available hardware on a robotic system. There is no possibility to directly adjust the model or incorporate any of the operator’s context knowledge. That is the motivation of the proposed DE-fine-ViT model: provide interactive interfaces that can adjust the model’s task understanding to improve performance and robustness, while still preserving the original capabilities of the DE-ViT model by making these interfaces optional and operating on top of the original architecture. Instead of relying on the automatic part discovery procedure (Section 4.4), the interfaces allow the operator to directly point the model to the characteristic attributes of a category by labeling them in the support set. Based on these annotations, the part-level prototypes are created and are then used by the FRM. The variant M2 represents the minimal upgrade compared to DE-ViT and, as can be seen in Table 6.13, improves its mAP performance on the fine-grained power drill dataset. Further performance can be gained by using more image samples per category with variant M3 or by making sure that the characteristic parts are visible from multiple

perspectives in the annotated samples (M4). It is important to mention that the bounding box of a prediction is not adjusted by the FRM. Thus, any improvements regarding the mAP come from correcting false category predictions. The effects of these corrections on precision and recall are listed in Table 6.14, supporting the observations that prototype multi-granularity, with both category- and part-level prototypes, and increasing the number of samples are the two main factors that can improve the fine-grained detection performance of the DE-fine-ViT model. However, choosing the parts for each category is not trivial and needs to be done based on feedback from the model. Example results are explained in the following.

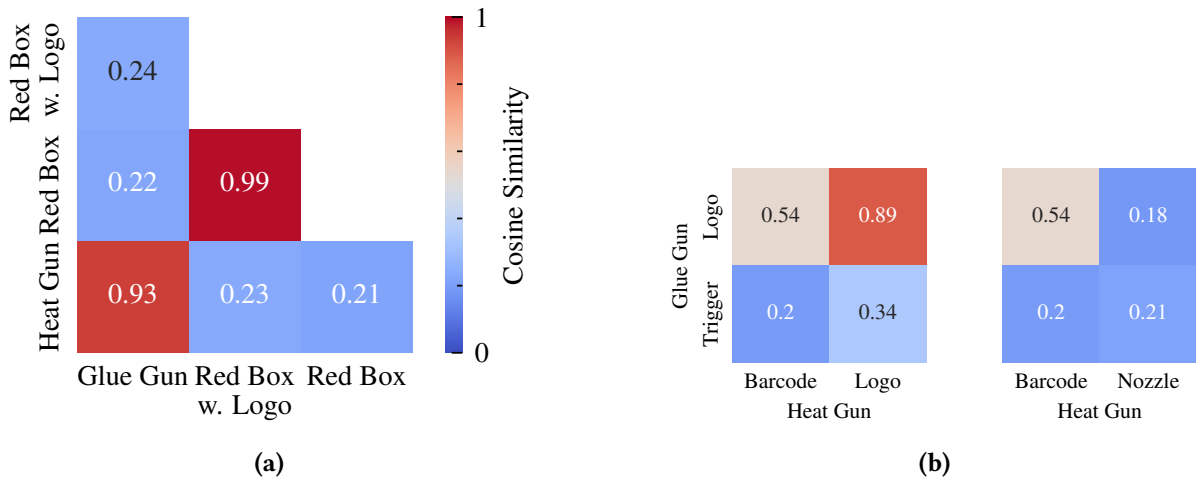


Figure 6.16: Similarity matrices for the mini-tool-dataset. (a) The matrix reveals two pairs of classes with strong visual similarity. As the class prototypes are fixed, distinctions must be enabled through parts. Similarity analysis of the part prototypes (b); before vs. after) for the visually similar classes heat gun and glue gun. The initial selection includes two visually similar logos (left). By replacing them, a set of more distinctive parts can be obtained, for example by introducing the nozzle component (right). Copyright ©2025, IEEE [P2].

Parts improve the fine-grained recognition capabilities of DE-fine-ViT, but they cannot be chosen arbitrarily. There are certain requirements that need to be fulfilled. Therefore, the part selection and annotation process is performed interactively based on feedback via similarity visualizations of the ViT backbone. An alternative is the automatic part discovery method described in Section 4.4. Since the descriptors of the backbone are used to determine the prototypes for categories and for parts, the backbone’s perception needs to be taken into account. In Fig. 6.16 the mini-tool-dataset, used for previous evaluation, is used as an example. If class prototypes of two categories are too similar Fig. 6.16a, the FRM is essential to help prevent confusions, but it needs part prototypes for that. If these part prototypes are too similar as well, confusions cannot reliably be resolved. So while selecting and annotating parts, the operator needs to be provided with the similarity matrices for class and part-level prototypes (Fig. 6.16). In Fig. 6.16b on the left, the similarity matrix for the initial parts chosen by the operator is shown. While the parts were perceived as different enough by the operator, they are quite similar, in terms of cosine similarity, for the ViT backbone. This would lead to an ineffective FRM. The operator then has the chance to change or add parts for the categories to ensure a part-similarity matrix where all cosine similarity values are as small as possible.

Note that category prototypes are fixed, so the matrix in Fig. 6.16a cannot be modified by any user interaction.

Automatic Part Discovery

As an alternative to the part prototypes created from user annotations, a basic baseline method for using the same architecture with automatically derived part prototypes (Section 4.4) is used for comparison. For this it clusters the patches within a bounding box using k-medoids [126]. Each cluster center becomes a part candidate which is assessed based on three scores: recurrence in the class-specific support images, discriminativeness with respect to part candidates of other similar classes, and spatial compactness of the corresponding cluster elements. To better compare the user-defined and the automatically discovered prototypes, a setup was chosen where the Region Proposal Network (RPN) and the detection head are replaced by simply using the ground-truth boxes, with a set of random predictions assigned including the correct category, as input for the part-based re-evaluation module.

Table 6.15 shows the results of a direct comparison of part prototypes on ground-truth input data, while the rest of the parameters and weights are completely identical. For the automatic part discovery the configuration was set to $N_{\text{parts}} = 4$, $\tau_{\text{rec}} = 0.55$, $\tau_{\text{dedup}} = 0.85$, and $N_{\text{cand}} = 6$. So while the number of parts per class was equal or lower in the user-labeled variant, the precision and recall were often superior to the automatically discovered part prototypes. Interestingly, the recall of the user-annotated parts for `drill_5` dropped, whereas the automatic version was approximately twice as good. On the contrary, it was the other way around for `drill_4`.

The advantage of an automatic discovery of parts is that it can be done without supervision, only the class-level bounding box annotations for the objects are required. The disadvantage is that the method does not have any context knowledge and solely assesses part candidates based on the data available in the support set. Which parts are relevant for the category represented by the few samples can only be assumed by the model. The three scores for recurrence, discriminativeness, and spatial compactness introduce the corresponding *biases* for part selection. The method is not elaborate and could certainly be made more complex. However, complexity cannot replace context knowledge that a human operator might have. A combination of both might be a suitable strategy for future research: the operator can freely define parts but is assisted by the automatic part discovery. The method could suggest high-scoring part candidates via a graphical user interface (GUI) to be chosen or declined by the operator, similar to ProtoPDebug [106].

In a restricted grid search within the parameter space of the automatic part discovery, an automatic configuration was found that decreased the number of false positives by around 50%, achieving a precision of 0.893 and a recall of 0.786, which surpasses the user-labeled part prototypes. However, it used $N_{\text{parts}} = 10$ instead of 4 parts per class, and therefore more than twice as many.

Table 6.15: Best-case per-class precision and recall on the fine-grained power drill dataset (61 ground-truth boxes, v2 annotations; Section A.4) using DE-fine-ViT’s re-evaluation module with top- k_{RoI} and ground-truth boxes as optimal predictions. *Labeled* uses the part prototypes created from user annotations, whereas *Auto* uses the automatic discovery procedure described in Section 4.4.

Class	Labeled			Auto			Δ (Auto – Labeled)	
	TP/FP/FN	Precision	Recall	TP/FP/FN	Precision	Recall	$\Delta\text{Prec.}$	$\Delta\text{Rec.}$
drill_1	12 / 9 / 1	0.571	0.923	10 / 2 / 3	0.833	0.769	+0.262	−0.154
drill_2	10 / 1 / 3	0.909	0.769	10 / 8 / 3	0.556	0.769	−0.354	0.000
drill_3	9 / 1 / 5	0.900	0.643	7 / 1 / 7	0.875	0.500	−0.025	−0.143
drill_4	10 / 4 / 0	0.714	1.000	2 / 2 / 8	0.500	0.200	−0.214	−0.800
drill_5	3 / 0 / 8	1.000	0.273	6 / 2 / 5	0.750	0.545	−0.250	+0.273
Mean	44 / 15 / 17	0.819	0.722	35 / 15 / 26	0.703	0.557	−0.116	−0.165

Multi-Layer Part Prototypes

In another experiment, it is examined how using multi-layer prototypes for the parts influences the mAP performance of the interactive model variant M4 on the fine-grained power drill dataset. The resulting mAP values for different depths of layers used for creating the part prototypes are depicted in Fig. 6.17. While multi-layer prototypes proved to be effective for the fine-grained recognition approach presented in Section 4.3 and evaluated in the previous section (Section 6.1.3), it can be seen that only using the last layer, which is the same depth used for class-level prototypes, performs best.

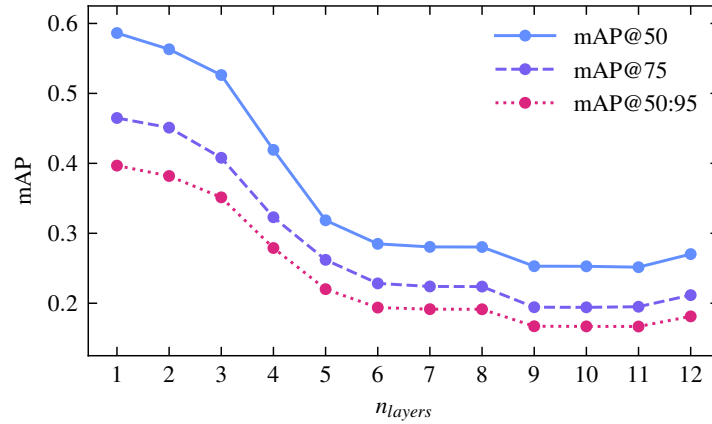


Figure 6.17: Detection performance of variant M3 of the proposed DE-fine-ViT model while using the last n_{layer} layers for part prototypes. Interestingly the performance slightly increases when using all 12 layers, indicating that the first layer (i.e., the 12th last layer) contains beneficial information. Copyright ©2025, IEEE [P2].

Conclusion

The presented approach extended the recently proposed DE-ViT model with multi-granular prototypes. Instead of only having class prototypes that represent a category and are used for classifying and regressing the region proposals, part-level prototypes are used to provide

a category description on a finer level. While these part-level prototypes can be selected automatically, the architecture also provides optional interfaces so that the operator can select characteristic parts of categories that cannot necessarily be inferred by the model due to the limited available data samples. This prototype multi-granularity is optional. Without adding part-level prototypes, the model behaves identically to the DE-ViT model. This means the rather high level of generalization caused by the underlying DINO-features can be reduced and adjusted to more specific subcategories. The experiments indicate that this can be beneficial for fine-grained tool detection, especially when visually similar categories differ in small but characteristic local regions. However, the options to do so are limited. Since the effectiveness of the parts relies on their unique visual appearance, functional differences such as different geometric shapes, for instance, are not feasible for refining target categories. The parts need to have strong and measurable differences in their extracted features, which are directly used for representing them. This also makes the distance to the object and the resulting object size in the image relevant. If the object is only represented by a small number of image patches, the local characteristics of the part prototypes can vanish or become too weak for a reliable re-evaluation. This is a strong limitation of the approach, since the perception of the backbone cannot be altered. Another limitation is that the relative positional arrangement of characteristic parts is currently not taken into account. However, since the same backbone type is used as in the part-based fine-grained recognition pipeline evaluated in the previous chapter, the structural information conveyed by the features could be used for at least coarsely assigning a certain structural part to the visual part prototype.

During inference, the parts help with explaining the decision of the model and can be used for downstream tasks, such as manipulating the object relative to the recognized parts. For instance, in a practical use case the chosen part on the handle was used for determining a pixel on the object which could be forwarded to a pixel-based grasp pipeline. The interactive preparation and inspection steps therefore add a useful interface between the operator's task knowledge and the model's prototype representation. At the same time, the quantitative analysis of this interaction is still limited. The experiments mainly show that the interfaces can support the intended workflow, but they do not yet provide a systematic measurement of how interaction quality, selected parts, or user decisions influence the final detection performance. Furthermore, the evaluation focuses on tool categories. This is suitable for the intended robotic retrieval scenario and provides an initial application-oriented validation, but only one object domain is examined quantitatively. Additional experiments with other tools and further robotic-related objects are required to assess the significance and generality of the observed improvements.

6.2 Results in Research Applications

This section shows a selection of research applications where the proposed architectures and methods were integrated. Note that it does not represent a thorough evaluation, but rather presents qualitative results that give an informative view of different scenarios where the architectures and methods have proven useful.

6.2.1 ROBDEKON

In the ROBDEKON¹ project the main focus was on decontamination scenarios with robots to reduce risks for humans. Multiple institutes and companies were part of it. Two of the work packages were relevant for this work. The first application was an emergency retrieval use case for potentially hazardous objects. For the object categories only a few images existed and the preparation time was assumed to be limited. A robot team collaborated to search for the target objects in the field, locate them, and store them safely (Fig. 6.18). Here, the instance-based data augmentation pipeline combined with the fine-tuning procedure from Section 4.2 was applied to quickly train the object detection model for the task at hand. Note that the optional depth fusion was not active during the search phase since the scenario did not require it. However, if there were multiple visually similar containers of hazardous objects of the same type but with different sizes and contents, the hybrid inference with depth fusion would allow the operator to quickly define rules based on the corresponding geometric dimensions. If the model suffered from too many false positives in a new mission environment, this was quickly resolved by adding images similar to the environment to the augmentation pipeline as *hard backgrounds*, so that they were included more frequently in the updated augmented dataset for the next fine-tuning. As can be seen in Fig. 6.19, together with the strategy of masking the robot itself, the detection model successfully detected a mock-up hazardous item on its 360° wide-angle camera without any false positives in a cluttered scene.

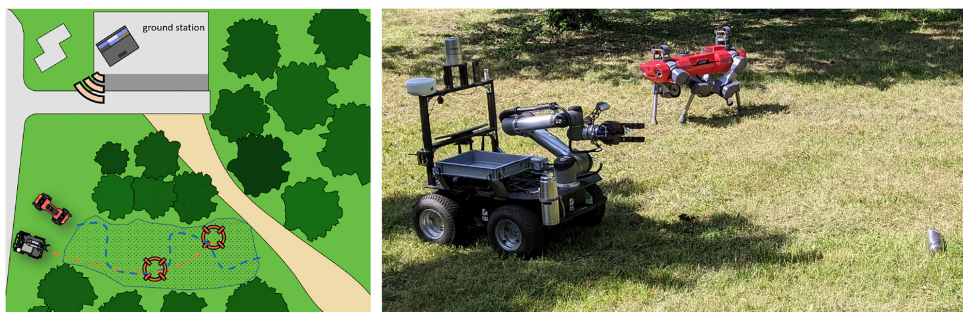


Figure 6.18: The mission environment is shown on the left. On the right, an image of the two mobile robots participating in the mission is displayed, along with the mock container for hazardous objects (bottom right). Image taken from [P8], licensed under CC BY 4.0.

¹ <https://robdekon.de>



Figure 6.19: Qualitative example of the detection of a hazardous container on a wide-angle camera of the mobile robot *Husky*. Although the environment is cluttered and contains many metal, cylindrical objects there are no false positives. Image taken from [P8], licensed under CC BY 4.0.

In another use case a recycling demonstrator was developed where the augmentation and depth fusion methods presented in Section 4.2 helped to create a robust battery sorting application. This demonstrator was also used as the evaluation scenario for some of the experiments presented in Section 6.1.2. More results and information about the ROBDEKON project can be found in [P7].

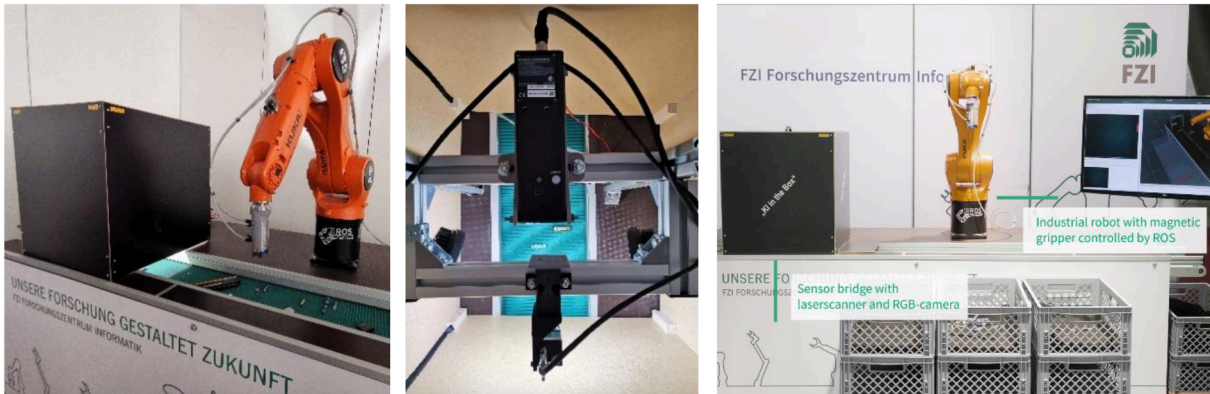


Figure 6.20: The battery sorting demonstrator used for evaluating the approach described in Section 4.2. On the right, the full demonstrator is depicted consisting of a sensor bridge (covered by a black box), the table with the integrated conveyor belt, the sorting boxes in the front and at the end of the conveyor belt, and the robot arm with a magnetic gripper. On the left a closer view of the conveyor belt and the robot arm is shown. The photo in the middle shows the sensor bridge with the industrial line scanner (top) and the high-speed RGB camera (bottom). Images from [P7].

6.2.2 ENRICH 2025

The ENRICH¹ hackathon is a competition at a nuclear plant in Austria that was never in operation. The goal is to solve various tasks ranging from rescuing persons to closing valves, either in autonomous or teleoperation mode. In the 2025 edition, the pipeline presented in Section 4.2 for fine-tuning pretrained models in time-critical missions was applied for rapid model preparation to detect orange hazard identification panels.

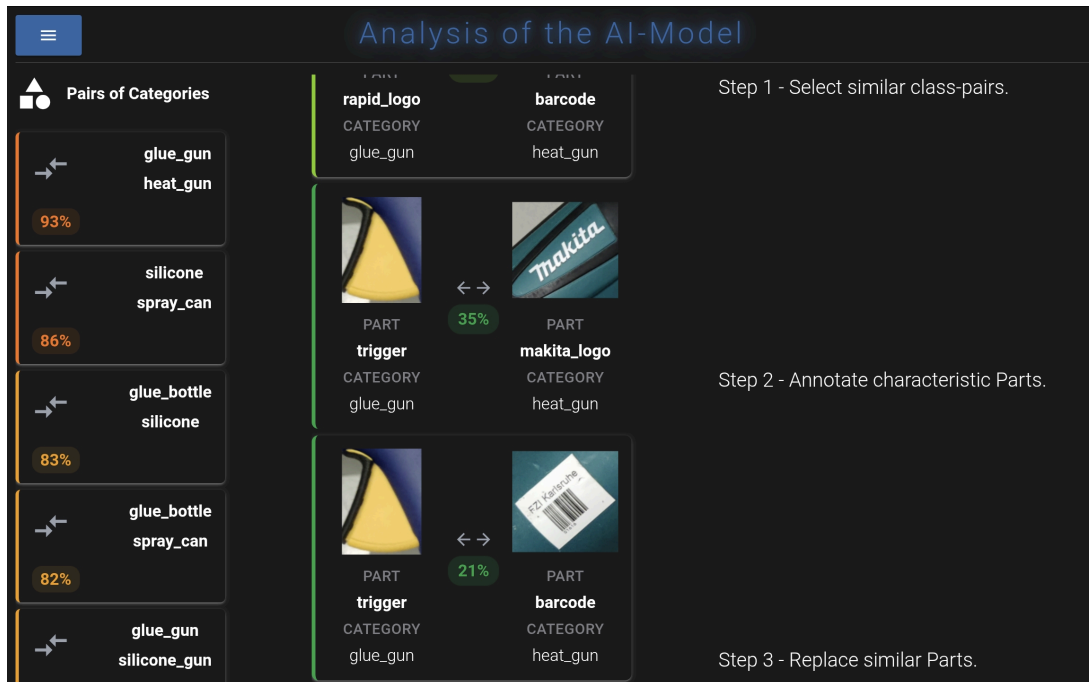


Figure 6.21: Screenshot of the browser-based user interface developed for the i-SAMLR project. The depicted panel allows the operator to see which categories are similar at the object level (left column), and after selecting the top pair (glue gun vs. heat gun) the part-level similarities are listed as well, together with example crops of the corresponding parts. A lower similarity increases robustness by avoiding confusions. Note that there are several more panels, e.g., for assisted annotation of objects and parts, or a scene overview where tasks for the robot can be commanded via speech.

6.2.3 i-SAMLR

The i-SAMLR project, financed by the Baden-Württemberg Stiftung², made the development of large parts of the architectures presented in Sections 4.3 and 4.4 possible. The targeted scenario was autonomous robots that assist outside of controlled environments with simple retrieve-and-bring tasks, e.g., on construction sites. To ensure flexibility, it must be possible to add objects to the perception system spontaneously during operation. The known categories can then be referenced within these assistive tasks. Thus, the goal was to create methods that enable training new objects online and on-site, and that are suitable for operators even

¹ <https://enrich.european-robotics.eu>

² <https://www.bwstiftung.de>

without any machine learning expertise. The final perception system is based on the DE-fine-ViT model ([Section 4.4](#)), which enables adding generic categories quickly using only a few samples, but also provides interfaces to guide the model towards characteristic features for a more fine-grained differentiation of similar categories. With this, common tool categories can be taught, such as power drill or screwdriver, but also fine-grained subcategories, e.g., only toolboxes with a certain company sticker. To reduce the complexity of using the model, a browser-based user interface (UI) was designed and connected with the application's ROS 2 stack. In [Fig. 6.21](#) a screenshot of the UI's panel for model analysis is depicted, where the similarities of category and part prototypes are visualized.

6.3 Discussion

This section discusses the findings of the combined evaluation, mainly presented in [Section 6.1](#), with respect to the methods described in [Chapters 4 and 5](#). The discussion relates the experimental results to the three research questions introduced in [Section 1.2](#) and assesses to what extent the proposed approaches support robotic perception under constrained resources. It also outlines the conditions under which these approaches are useful and where their limitations remain.

It is common practice to pre-train perception models and then fine-tune them on a smaller, task-specific dataset. The distinctive aspect of this work is the application of this principle to robotic perception together with adaptations that either avoid changing the pretrained backbone or restrict task-specific learning to lightweight components. For a more in-depth methodological analysis, see the conclusions of the respective evaluation sections.

Regarding the first research question **RQ1**, the experimental results show that vision models pretrained on general datasets can be adapted to specific object-related perception tasks using different strategies. In some cases, these tasks differed significantly from those addressed during pretraining. Across the evaluated architectures, preparation effort is reduced mainly by avoiding extensive task-specific datasets. This is achieved explicitly through data augmentation methods and implicitly by exploiting depth information or the part-aware representations already present in the backbones. These choices also reduce preparation time, since the developed architectures require only short fine-tuning phases and in some cases no fine-tuning at all. They also lower the demand for hardware such as VRAM (video random access memory). VRAM becomes especially important when all parameters of large models are optimized with gradient-based methods. In this work, however, usually only a few layers are optimized, and some preparation steps require only forward passes through the models. These forward passes typically need the same amount of VRAM as later inference. Fewer preparation steps also tend to reduce the amount of machine learning experience required from the operator. A precise quantification of user expertise was, however, not part of this work. Data and hardware requirements are therefore reduced directly, while time and expertise are addressed only indirectly.

An important observation is that the internal features of pretrained models are usually meaningful enough to generalize to previously unseen object categories, even in domain-specific scenarios such as waste or tool recognition. This supports the assumption that large-scale pre-training, even without supervision, can produce transferable visual embeddings that remain useful with only minor architectural additions. The evaluations also show, however, that this benefit depends on whether the task-relevant differences are represented in these embeddings.

This dependency is central to the second research question **RQ2**. The evaluated architectures covered both tasks with high intra-class variability and tasks with high inter-class similarity. For high intra-class variability, the proposed approaches mainly relied on broad generalization followed by mechanisms for reducing false positives, either through known *false* categories

or through geometric shape analysis. This worked in the examined settings, but it should not be interpreted as a general solution for all highly variable object categories. Many robotic applications involve forms of variability for which neither expected distractor categories nor simple geometric verification are available.

For high inter-class similarity, the proposed part-based recognition and multi-granular prototype mechanisms showed that fine-grained distinctions are possible when the relevant differences are visible to the pretrained backbone. At the same time, these results define an important limit of the approach. The architectures can only use features that are encoded in the model representation and can be addressed through the interfaces. These features usually need to be simple and visually distinctive, such as a characteristic color or a salient object part. Shape-based differences, e.g., a triangular sticker versus a circular one, are not always represented distinctly enough by the model to support reliable differentiation. Furthermore, text-based features such as logos or labels with discriminative text are not yet exploited for fine-grained recognition, despite their promising potential.

In general, more data is helpful when it covers the variability of the categories more completely. The proposed approaches should therefore not be seen as replacements for fully trained models when sufficient data, time, and hardware are available. Their role is to provide suitable alternatives when data is scarce or when collecting and annotating larger amounts for a specific robotics application on short notice would be too costly or time-consuming.

The third research question **RQ3** addressed whether operator knowledge can improve task alignment under such constraints. The two architectures that focus on fine-grained vision tasks show why this is relevant. With limited data, it is often underspecified which features or parts define the intended category. The VINTER model (Sections 4.3 and 5.1) relies on a part-oriented inductive bias based on visual matching in its non-interactive mode. To incorporate context knowledge, it also provides user interfaces for changing the reference prototypes of each part. Similarly, the interactive interfaces of the DE-fine-ViT architecture (Section 5.2) offer a way to adjust model behavior when the operator knows or anticipates which visual aspects are relevant for the task. This can improve task alignment without collecting additional samples. The same mechanism also exposes a practical limitation. Images of expected false positives are often needed to verify them and to counteract them effectively through the interfaces. If such examples are not available, the category-specific generalization of the model can only be validated roughly. Moreover, the evaluation does not yet quantify the required user effort or the variability introduced by different operators. The results therefore show the technical feasibility of interaction, but not yet its full usability or reproducibility in a broader deployment setting.

Overall, the results support a positive answer to the first research question **RQ1**: pretrained models can be integrated into new architectures using lightweight additional modules and adapted for specific perception tasks. This is backed by evaluations on different robotic applications. For **RQ2**, the answer is more conditional. The experiments indicate that few-shot

methods without extensive fine-tuning are more reliable for tasks with low intra-class variability. For fine-grained settings with high inter-class similarity, success depends on the features that separate the categories. If the pretrained backbone is sensitive to these features, as shown with the DE-fine-ViT architecture (Section 4.4), fine-grained differentiation can be improved through part-level prototypes. If the relevant distinction is not reflected strongly enough in the backbone features, e.g., because it is too small, textual, or shape-based, the current architectures cannot reliably compensate for this. For **RQ3**, the proposed interfaces provide a technically viable means of integrating operator knowledge into fine-grained perception tasks. They are less suitable for refining generic categories, since they mainly target part-level attributes on a less abstract level, such as colors and textures. Overall, the evaluation supports the use of pretrained backbones, lightweight adaptation, and interaction as complementary tools for resource-constrained robotic perception, but only under task conditions where the required visual evidence is present in the model representation and can therefore be inspected or adjusted interactively with the proposed methods.

7 Conclusion and Future Work

This chapter concludes the research presented in this work. The different methods and approaches are summarized. The findings and contributions are discussed and assessed. Finally, future research directions are motivated.

7.1 Summary

The overarching research objective of this work addresses a central challenge in robotic visual perception: enabling object-centric recognition in scenarios where large labeled datasets are unavailable and extensive training procedures must be avoided. The developed methods share the common strategy of leveraging the inherent capabilities of pretrained models with frozen parameters for the tasks at hand. In addition, lightweight architectural extensions are proposed for a diverse set of perception tasks, several of which operate with frozen backbones and require no fine-tuning. The range of tasks is characterized by different degrees of intra-class variability and inter-class similarity ([Section 3.1.1](#)).

Starting with the repurposing of attention maps of pretrained vision transformers for object segmentation, it was investigated whether pretrained models can be used directly, without adjusting any parameters, for tasks they were not trained for: litter segmentation. Another part of this work targets a comparable challenge with similarly high intra-class variability. When data scarcity prevents the training samples from accurately capturing the true intra-class variability, a transfer learning-based approach was proposed that additionally processes depth data to avoid false positives caused by intended over-generalization, especially when visual features alone are not sufficient for category differentiation but geometric dimensions are. Moving on to scenarios with high inter-class similarity, a recently uncovered semantic structural understanding of self-supervised feature extractors in the DINO family [\[36\]](#) was used in a part-based object recognition pipeline. Analogously, multi-granularity feature-based prototypes were employed in a few-shot object detection scenario, in which an existing, strong-performing architecture [\[88\]](#) was extended with lightweight modules to acquire fine-grained differentiation capabilities. Beyond this, a further chapter addressed the question of whether interactive learning, also known as human-in-the-loop, is beneficial in such scenarios with fine-grained tasks and similar categories. User interfaces were integrated into the proposed architectures and preparation procedures. For instance, the idea of concept banks [\[51\]](#) as translators between the human and the perception model was adapted. When data is limited and the model's inductive biases pick up spurious features, such interfaces facilitate corrective feedback. Furthermore, a concept of shared responsibility between the model and the robotic

operator was developed for selecting suitable characteristic features during the preparation phase of the multi-granular object detection architecture mentioned above.

While the different approaches perform distinct visual tasks, such as segmentation, recognition, or detection, and target different applications, they provide insights into the challenging conditions of high intra-class variability, high inter-class similarity, and data scarcity that are relevant for developing next-generation models. Combining multiple of these specialized capabilities could allow models to adapt more readily to the varying task demands encountered during deployment. This is an advantage for robotic applications, where missions can be spontaneous or time-critical. Alongside the methodological contributions, a curated survey of perception-related datasets relevant to robotic applications is provided in the appendix, intended as a resource for evaluating few-shot perception methods on less commonly used data.

7.2 Conclusion

This work addressed the challenge of enabling object-centric visual perception for robotic applications under constrained resources. With respect to the first research question, how pretrained vision models can be adapted to specific tasks while reducing preparation requirements, the results of this dissertation demonstrate that direct reductions in data and compute hardware, as well as indirect reductions in expertise and time, are achievable. Across multiple perception tasks, including segmentation, recognition, and detection, pretrained models were successfully repurposed without requiring extensive training. By leveraging frozen pretrained models and their corresponding feature representations, the introduced lightweight adaptation strategies, such as modular plain processing pipelines and part-based reasoning, reduce the need for large-scale dataset creation and computationally demanding parameter optimization. This confirms that pretrained models provide transferable visual understanding that can be effectively exploited in robotic applications. At the same time, the results show that the success of these approaches depends on the concrete conditions of the target task.

The second research question examined whether few-shot learning architectures can achieve robust performance across varying levels of task granularity. Existing state-of-the-art few-shot models are already effective in coarse-grained scenarios, where categories differ significantly in visual appearance and thus exhibit lower inter-class similarity. In these cases, the feature embeddings provided by the pretrained models are sufficiently discriminative to enable reliable recognition with only a few examples. As the level of granularity becomes finer, the limitations of single-level representations from pretrained models become more evident. To address this, the proposed methods introduce part-based and multi-granular representations that enable the systems to focus on low-level and more discriminative features for the task at hand. These extensions improve performance in fine-grained scenarios without fine-tuning the underlying backbones. The findings suggest that while few-shot methods can be extended to finer-grained levels, their effectiveness is bounded by the representational granularity of

the underlying feature space. Fine-grained distinctions become difficult when concepts that are distinct to humans are mapped to similar feature representations by the model.

The third research question focused on integrating operator knowledge through interactive mechanisms to enhance task alignment without requiring additional data. The set of presented methods demonstrates that human interaction can be a valuable source of information in data-scarce settings. By comprehensively presenting intermediate model representations and enabling targeted adjustments, such as emphasizing relevant visual concepts or guiding attention toward specific object parts, operators can directly influence the inference process. This allows the system to interactively incorporate contextual user knowledge that cannot be inferred solely from the limited training samples. The results indicate that such interaction can lead to improvements, particularly in fine-grained tasks where purely data-driven approaches suffer from underspecification or shortcut learning, although a systematic quantitative comparison against non-interactive baselines remains outstanding. However, the effectiveness depends heavily on whether the underlying representation captures the visual distinctions required to separate the categories in the given task. While the proposed interfaces also aim to minimize the required user expertise, empirical observations indicate that the interactive preparation procedures are not yet intuitive enough and require operator training. It should also be noted that the quantitative evaluation of the interactive procedures is limited in scope. Broader user studies with more participants and tasks would be needed to establish the statistical significance of the observed benefits. Despite this, integrating operator knowledge is a promising alternative to purely data-driven adaptation of perception models.

The contributions of this dissertation establish a framework for resource-efficient robotic perception that combines the exploitation of pretrained models, lightweight modular model components, and optional interactive refinement. The methods demonstrate that it is possible to reduce dependence on data-intensive training pipelines, though not without trade-offs and limitations. A combination of strong pretrained models, *shallow* task-specific modules, and operator feedback can help meet the challenging constraints of real-world robotic applications, as illustrated by the integration of the proposed methods into the research projects summarized in [Section 6.2](#). This is particularly relevant for scenarios involving rare objects, limited onboard hardware capabilities, or time-critical mission preparation. At the same time, less data and less training always impose trade-offs, as they restrict the degree of task specification and model adaptability. The reliance on pretrained models and the representations they produce introduces dependencies on their generalization capabilities. However, new backbones are constantly developed and published, and can substitute the current ones with little effort across all architectures, potentially leading to direct performance gains.

In conclusion, this work demonstrates that practical visual perception for robotics can be achieved with reduced resource requirements across diverse scenarios. The presented methods are intended to provide a starting point for subsequent research in the respective areas. A detailed technical discussion of the findings is given in [Section 6.3](#).

7.3 Perspectives for Further Research

The architectures and findings of this dissertation open promising directions for future research in robotic perception. While the presented methods demonstrate several use cases in which pretrained models and interactive mechanisms can reduce preparation requirements, they also reveal limitations that need to be addressed to improve practical usability.

Future work could further investigate the integration of intermediate adaptation strategies, such as parameter-efficient fine-tuning, that retain the efficiency of the proposed methods while increasing flexibility and potentially robustness. In particular, approaches that selectively adapt only small subsets of model parameters may provide a suitable compromise. However, replacing fine-tuning strategies with foundation models that offer broader contextual knowledge and a stronger intuition for object categories appears more promising.

The integration of human interaction offers significant potential for further exploration. While this work demonstrates how operator guidance can be beneficial in compensating for limited data, the design of intuitive and scalable interfaces remains an open challenge. Future research could aim to reduce the user’s cognitive load, for instance by automatically suggesting characteristic attributes that guide users toward the most impactful interactions. Another promising direction lies in generative models that iteratively explore the intended decision boundaries by synthesizing edge-case samples.

A broader direction is to integrate the proposed strategies more tightly into a unified perception framework. For example, the part-based prototype functionality of the VINTER model (Sections 4.3 and 5.1) could be used to extend the DE-fine-ViT model (Sections 4.4 and 5.2) with a concept bank for color- or texture-related adjustments of the category prototypes. This merging of modular components could be further supported by integrating and testing the most recent backbone variants, e.g., DINOv3 [137] or the promising Perception Encoder [138].

Future work should also reconsider the region proposal stage used in many two-stage few-shot detection pipelines. Region Proposal Networks (RPNs) are efficient, but their objectness estimations are learned from the training distribution and can therefore be weak for novel, highly specific objects or objects from completely different domains, such as underwater or aerial scenarios [100]. This limitation is directly relevant for few-shot robotic detection, where missed or imprecise proposals cannot be recovered by later prototype matching. Recent methods that combine class-agnostic segmentation with DINO-based feature matching point to an alternative direction. For instance, the training-free reference-based instance segmentation approach by Espinosa *et al.* [139] uses DINOv2 and SAM 2 correspondences to generate instance masks from a small set of reference images. Such proposal mechanisms could be further investigated as replacements for RPNs when the target objects are highly specific and insufficiently represented in the RPN’s training distribution. From a critical perspective, however, segmentation foundation models such as SAM are not immune to distributional bias either.

Although trained on substantially broader data than typical RPNs, they may equally underperform on unusual object categories. Thus, their advantage over RPNs in this regard remains to be systematically characterized.

As onboard hardware and embedded Graphics Processing Units (GPUs) become increasingly powerful, performing near-real-time inference with vision-language models (VLMs) locally on the robot becomes more feasible. Recent open-concept segmentation models, such as SAM3 proposed by Carion *et al.* [130], provide improved qualitative results and represent a promising intermediate solution. By allowing categories to be specified through complex textual descriptions, they enable finer-grained distinctions than open-vocabulary detectors. However, language alone is often not expressive enough to precisely describe objects or vision tasks to a model. Instead, textual semantic task descriptions grounded with example images, combined with a model that comprehends both domains, could lead to a powerful perception system.

7.4 Lessons Learned

A notable challenge during this work was the rapidly evolving landscape of vision models. The continuous emergence of stronger architectures, particularly vision transformers and foundation models, repeatedly shifted the baseline of what could be achieved without extensive task-specific training. As a result, earlier design decisions had to be revisited because the underlying capabilities of pretrained models had significantly improved. At the same time, this is also an advantage of incorporating pretrained models, as newer versions can lead to performance gains without changing any other components of the architectures. It was always challenging to decide whether to commit to a particular backbone early on and complete first versions of the architectures, or to directly examine and adapt to newly published models.

Regarding the software stack, while frameworks such as PyTorch¹ enable a degree of model abstraction, significant differences nevertheless remained across implementations, evaluation setups and metrics, as well as data and annotation formats. In addition, the often intricate package dependencies of both pretrained backbones and state-of-the-art few-shot models posed, at least initially, a recurring maintenance burden during development and evaluation. Since many few-shot models adapt and build on previous ones, their implementations are often reused as well, which explains why outdated package dependencies tend to persist for so long.

¹ <https://pytorch.org>

Own Publications

This collection contains a complete list of the **author**'s publications produced during the doctoral studies. Directly resulting from the research of this work are completely or in parts the publications [P2], [P1], [P3], [P5], [P7], [P8]. Other publications that are only indirectly related to the presented research are [P4], [P6], [P9]. The two publications on ROS 2 real-time performance [P10], [P11] are not related to the main topic nor visual perception in general. Nevertheless, in this work, ROS 1/ROS 2 has been used as the underlying framework for all practical experiments that involve robots. The entries are listed in descending order by year.

- [P1] **KELLER, P.**; MÖCK, M.; HEPPNER, G. and DILLMANN, R.: “Low-Compute Trash Segmentation Using Pretrained Vision Transformers Without Fine-Tuning,” in *2025 11th International Conference on Automation, Robotics, and Applications (ICARA)*, Feb. 2025, pp. 76–80. DOI: [10.1109/ICARA64554.2025.10977695](https://doi.org/10.1109/ICARA64554.2025.10977695) Accessed: May 8, 2025. [Online]. Available: <https://ieeexplore.ieee.org/document/10977695/> (cit. on pp. 41, 42, 86–88, 91, 131).
- [P2] **KELLER, P.**; STRECKER, L.; DURCHDEWALD, F.; GRAAF, F.; SCHNELL, T. and DILLMANN, R.: “Interactive Fine-grained Few-shot Detection of Tools,” in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2025, pp. 15 135–15 142. DOI: [10.1109/IROS60139.2025.11246465](https://doi.org/10.1109/IROS60139.2025.11246465) Accessed: Dec. 29, 2025. [Online]. Available: <https://ieeexplore.ieee.org/document/11246465/> (cit. on pp. 60, 63, 78, 109–114, 116, 131).
- [P3] **KELLER, P.**; JOST, D.; ROENNAU, A. and DILLMANN, R.: “Interactive Teaching For Fine-Granular Few-Shot Object Recognition Using Vision Transformers,” in *2024 IEEE International Conference on Image Processing (ICIP)*, Oct. 2024, pp. 535–541. DOI: [10.1109/ICIP51287.2024.10647556](https://doi.org/10.1109/ICIP51287.2024.10647556) Accessed: Jan. 10, 2025. [Online]. Available: <https://ieeexplore.ieee.org/document/10647556/> (cit. on pp. 53, 56, 60, 73, 99, 100, 102, 105, 106, 131).
- [P4] **KELLER, P.**; RIVERA, G. P.; SCHNELL, T. and DILLMANN, R.: “Online Vision-Based 3D Reconstruction for Shared Human-Robot Workspaces,” in *2024 4th International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, Nov. 2024, pp. 01–07. DOI: [10.1109/ICECCME62383.2024.10797153](https://doi.org/10.1109/ICECCME62383.2024.10797153) Accessed: Jan. 10, 2025. [Online]. Available: <https://ieeexplore.ieee.org/document/10797153/> (cit. on p. 131).
- [P5] **KELLER, P.**; MANGLER, J.; HÜGEL, N.; BESSELMANN, M. G.; RÖNNAU, A. and DILLMANN, R.: “Robotic Sorting of Batteries Using Visual Few-shot Learning and Fusion with Depth Data,” in *2023 3rd International Conference on Electrical, Computer,*

- Communications and Mechatronics Engineering (ICECCME)*, Jul. 2023, pp. 1–7. DOI: [10.1109/ICECCME57830.2023.10253347](https://doi.org/10.1109/ICECCME57830.2023.10253347) Accessed: Jan. 10, 2025. [Online]. Available: <https://ieeexplore.ieee.org/document/10253347/> (cit. on pp. 46, 48, 94–97, 131, 164).
- [P6] **KELLER, P.**; WENG, B.; WILBRANDT, R.; GRAAF, F.; ROENNAU, A. and DILLMANN, R.: “Intuitive Access to Versatile Assistive Services Provided by Agile Walking Robots Using Open Source LLMs,” in *IROS Workshop on Assistive Robotics for Citizens*, Oct. 5, 2023. Accessed: Jan. 10, 2025. [Online]. Available: <https://drive.google.com/file/d/1JzyzgvgRz0C0sQnbJlCnEIGaX5K6ErEm/view> (cit. on p. 131).
- [P7] MANGLER, J.; GROßE-BESSELMANN, M.; **KELLER, P.**; HÜGEL, N.; ROENNAU, A. and INFORMATIK, F. F.: “ROBDEKON - Teilvorhaben: Autonome, mobile Roboter für die Dekontamination in schwierigen Umgebungen : Schlussbericht ; im Rahmen des Programms ”Forschung für zivile Sicherheit”,” 2023. DOI: [10.2314/KXP:1880638576](https://doi.org/10.2314/KXP:1880638576) [Online]. Available: <https://www.tib.eu/en/search/id/TIBKAT:1880638576> (cit. on pp. 46, 94, 95, 119, 131).
- [P8] ROENNAU, A.; MANGLER, J.; **KELLER, P.**; BESSELMANN, M. G.; HUEGEL, N. and DILLMANN, R., “Grasping and retrieving unknown hazardous objects with a mobile manipulator,” *at - Automatisierungstechnik*, vol. 70, no. 10, pp. 838–849, Oct. 1, 2022. DOI: [10.1515/auto-2022-0061](https://doi.org/10.1515/auto-2022-0061) Accessed: Jan. 10, 2025. [Online]. Available: <https://www.degruyter.com/document/doi/10.1515/auto-2022-0061/html> (cit. on pp. 46, 94, 118, 119, 131).
- [P9] TANEV, A.; TIECK, J. C. V.; HEPPNER, G.; **KELLER, P.**; ROENNAU, A. and DILLMANN, R.: “Embedded AI-Box for Visual Detection and Inspection with Mobile Robots in ROS2,” in *Proceedings of WCSE 2022 Spring Event: 2022 9th International Conference on Industrial Engineering and Applications*, WCSE, 2022. DOI: [10.18178/wcse.2022.04.054](https://doi.org/10.18178/wcse.2022.04.054) Accessed: Jan. 10, 2025. [Online]. Available: <http://www.wcse.org/WCSE%202022%20Spring/054.pdf> (cit. on p. 131).
- [P10] PUCK, L.; **KELLER, P.**; SCHNELL, T.; PLASBERG, C.; TANEV, A.; HEPPNER, G.; ROENNAU, A. and DILLMANN, R.: “Performance Evaluation of Real-Time ROS2 Robotic Control in a Time-Synchronized Distributed Network,” in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, Aug. 2021, pp. 1670–1676. DOI: [10.1109/CASE49439.2021.9551447](https://doi.org/10.1109/CASE49439.2021.9551447) Accessed: Jan. 10, 2025. [Online]. Available: <https://ieeexplore.ieee.org/document/9551447/> (cit. on p. 131).
- [P11] PUCK, L.; **KELLER, P.**; SCHNELL, T.; PLASBERG, C.; TANEV, A.; HEPPNER, G.; ROENNAU, A. and DILLMANN, R.: “Distributed and Synchronized Setup towards Real-Time Robotic Control using ROS2 on Linux,” in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, Aug. 2020, pp. 1287–1293. DOI: [10.1109/CASE48305.2020.9217010](https://doi.org/10.1109/CASE48305.2020.9217010) Accessed: Jan. 10, 2025. [Online]. Available: <https://ieeexplore.ieee.org/document/9217010/> (cit. on p. 131).

Supervised Student Theses

The following collection lists student theses proposed and supervised by the **author** as part of his research. The underlying research questions and methodological foundations were mostly developed by the **author**, while the students contributed predominantly to their implementation and refinement. Selected results from these theses are incorporated into and extended by the research presented in this dissertation. The entries are listed in descending order by year.

- [T1] RIESTERER, M. and **KELLER, P.**: “Evaluating Few-Shot Object Detection Models in Robotic Tasks with Mixed Granularity,” Bachelor’s Thesis, Karlsruhe Institute of Technology, Jul. 15, 2025.
- [T2] SCHMETKAMP, I. and **KELLER, P.**: “Autonomous Object Inspection with Mobile Robots for 3D Reconstruction and Image Data Acquisition,” Bachelor’s Thesis, Karlsruhe Institute of Technology, Mar. 17, 2025.
- [T3] DEITERSEN, M. and **KELLER, P.**: “Interaction with Models for Few-shot Object Detection,” Seminar Paper, Karlsruhe Institute of Technology, Mar. 2, 2024.
- [T4] DURCHDEWALD, F. and **KELLER, P.**: “Evaluation of Few-shot Object Detection Approaches for Mobile Robots,” Bachelor’s Thesis, Karlsruhe Institute of Technology, Aug. 18, 2024 (cit. on pp. [109](#), [168](#)).
- [T5] STRECKER, L. and **KELLER, P.**: “Explainable and Interactive Few-Shot Object Detection for Autonomous Mobile Robots,” Master’s Thesis, Karlsruhe University of Applied Sciences, Sep. 30, 2024 (cit. on pp. [78](#), [109](#), [111](#)).
- [T6] JOST, D. and **KELLER, P.**: “Interactive Teaching of Unknown Target Objects for Mobile Robots,” Master’s Thesis, Karlsruhe Institute of Technology, Sep. 28, 2023 (cit. on pp. [53](#), [73](#), [99](#), [105](#)).
- [T7] WENG, B. and **KELLER, P.**: “Intelligent Speech-based Interaction Interface for Mobile Robots,” Bachelor’s Thesis, Karlsruhe Institute of Technology, Nov. 2, 2023.
- [T8] MÖCK, M. and **KELLER, P.**: “Attention-based Anomaly Segmentation for Trash Detection,” Master’s Thesis, Karlsruhe Institute of Technology, May 1, 2021 (cit. on pp. [41](#), [86](#), [88](#), [92](#)).
- [T9] BOSCH, C. and **KELLER, P.**: “Visual few-shot learning for mobile robots,” Bachelor’s Thesis, Karlsruhe Institute of Technology, Aug. 31, 2020.
- [T10] EJJALILI, A. and **KELLER, P.**: “Dataset-oriented design of neural networks for object detection,” Bachelor’s Thesis, Karlsruhe Institute of Technology, Apr. 17, 2020.
- [T11] XIAO, Z. and **KELLER, P.**: “Active Learning for Instance Segmentation in Conveyor Belt Applications,” Master’s Thesis, Karlsruhe Institute of Technology, Aug. 31, 2020.

Bibliography

- [1] LECUN, Y.; BOTTOU, L.; BENGIO, Y. and HAFFNER, P., “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998. DOI: [10.1109/5.726791](https://doi.org/10.1109/5.726791) Accessed: Dec. 30, 2025. [Online]. Available: <https://ieeexplore.ieee.org/document/726791/> (cit. on pp. 9, 13).
- [2] LECUN, Y.; BOSER, B.; DENKER, J. S.; HENDERSON, D.; HOWARD, R. E.; HUBBARD, W. and JACKEL, L. D., “Backpropagation Applied to Handwritten Zip Code Recognition,” *Neural Computation*, vol. 1, no. 4, pp. 541–551, Dec. 1989. DOI: [10.1162/neco.1989.1.4.541](https://doi.org/10.1162/neco.1989.1.4.541) Accessed: Dec. 30, 2025. [Online]. Available: <https://ieeexplore.ieee.org/document/6795724> (cit. on p. 9).
- [3] RUMELHART, D. E.; HINTON, G. E. and WILLIAMS, R. J., “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986. DOI: [10.1038/323533a0](https://doi.org/10.1038/323533a0) Accessed: Jan. 5, 2026. [Online]. Available: <https://www.nature.com/articles/323533a0> (cit. on p. 9).
- [4] RUSSAKOVSKY, O. et al., “ImageNet Large Scale Visual Recognition Challenge,” *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, Dec. 1, 2015. DOI: [10.1007/s11263-015-0816-y](https://doi.org/10.1007/s11263-015-0816-y) Accessed: Jan. 6, 2026. [Online]. Available: <https://doi.org/10.1007/s11263-015-0816-y> (cit. on pp. 10, 16, 28, 32, 34–36, 42, 86).
- [5] DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K. and FEI-FEI, L.: “ImageNet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp. 248–255. DOI: [10.1109/CVPR.2009.5206848](https://doi.org/10.1109/CVPR.2009.5206848) Accessed: Jan. 6, 2026. [Online]. Available: <https://ieeexplore.ieee.org/document/5206848> (cit. on pp. 10, 13, 86).
- [6] KRIZHEVSKY, A.; SUTSKEVER, I. and HINTON, G. E.: “ImageNet Classification with Deep Convolutional Neural Networks,” in *Advances in Neural Information Processing Systems*, vol. 25, Curran Associates, Inc., 2012. Accessed: Dec. 30, 2025. [Online]. Available: <https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html> (cit. on p. 10).
- [7] SIMONYAN, K. and ZISSERMAN, A. “Very Deep Convolutional Networks for Large-Scale Image Recognition.” arXiv: [1409.1556](https://arxiv.org/abs/1409.1556), Accessed: Oct. 21, 2024. [Online]. Available: <http://arxiv.org/abs/1409.1556> pre-published (cit. on pp. 10, 36).
- [8] IOFFE, S. and SZEGEDY, C.: “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” in *Proceedings of the 32nd International Conference on Machine Learning*, PMLR, Jun. 1, 2015, pp. 448–456. Accessed: Apr. 16,

2026. [Online]. Available: <https://proceedings.mlr.press/v37/ioffe15.html> (cit. on p. 10).
- [9] HE, K.; ZHANG, X.; REN, S. and SUN, J.: “Deep Residual Learning for Image Recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016, pp. 770–778. DOI: [10.1109/CVPR.2016.90](https://doi.org/10.1109/CVPR.2016.90) Accessed: Oct. 21, 2024. [Online]. Available: <http://ieeexplore.ieee.org/document/7780459/> (cit. on pp. 10, 36).
- [10] KRIZHEVSKY, A.: “Learning Multiple Layers of Features from Tiny Images,” 2009. [Online]. Available: <https://www.cs.utoronto.ca/~kriz/learning-features-2009-TR.pdf> (cit. on pp. 10, 13, 33).
- [11] VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, Ł. and POLOSUKHIN, I.: “Attention is All you Need,” in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017. Accessed: Sep. 15, 2022. [Online]. Available: <https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html> (cit. on p. 10).
- [12] DOSOVITSKIY, A. et al.: “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale,” presented at the International Conference on Learning Representations, Jan. 12, 2021. Accessed: May 24, 2023. [Online]. Available: <https://openreview.net/forum?id=YicbFdNTTy> (cit. on pp. 11, 18, 41, 42, 86).
- [13] LIU, Z.; LIN, Y.; CAO, Y.; HU, H.; WEI, Y.; ZHANG, Z.; LIN, S. and GUO, B.: “Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows,” presented at the Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 10 012–10 022. Accessed: Apr. 16, 2026. [Online]. Available: https://openaccess.thecvf.com/content/ICCV2021/html/Liu_Swin_Transformer_Hierarchical_Vision_Transformer_Using_Shifted_Windows_ICCV_2021_paper (cit. on p. 11).
- [14] LIU, Z.; MAO, H.; WU, C.-Y.; FEICHTENHOFER, C.; DARRELL, T. and XIE, S.: “A ConvNet for the 2020s,” presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 11 976–11 986. Accessed: Apr. 16, 2026. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2022/html/Liu_A_ConvNet_for_the_2020s_CVPR_2022_paper.html (cit. on p. 11).
- [15] REN, S.; HE, K.; GIRSHICK, R. and SUN, J.: “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” in *Advances in Neural Information Processing Systems*, vol. 28, Curran Associates, Inc., 2015. Accessed: Mar. 3, 2023. [Online]. Available: <https://papers.nips.cc/paper/2015/hash/14bfa6bb14875e45bba028a21ed38046-Abstract.html> (cit. on pp. 11, 12, 29, 36).
- [16] UIJLINGS, J. R. R.; van de SANDE, K. E. A.; GEVERS, T. and SMEULDERS, A. W. M., “Selective Search for Object Recognition,” *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, Sep. 1, 2013. DOI: [10.1007/s11263-013-0620-5](https://doi.org/10.1007/s11263-013-0620-5) Accessed: May 26, 2026. [Online]. Available: <https://doi.org/10.1007/s11263-013-0620-5> (cit. on p. 11).

-
- [17] GIRSHICK, R.; DONAHUE, J.; DARRELL, T. and MALIK, J.: “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,” presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 580–587. Accessed: Dec. 29, 2025. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2014/html/Girshick_Rich_Feature_Hierarchies_2014_CVPR_paper.html (cit. on pp. 12, 16).
- [18] GIRSHICK, R.: “Fast R-CNN,” presented at the Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1440–1448. Accessed: Apr. 16, 2026. [Online]. Available: https://openaccess.thecvf.com/content_iccv_2015/html/Girshick_Fast_R-CNN_ICCV_2015_paper.html (cit. on p. 12).
- [19] LIU, W.; ANGUELOV, D.; ERHAN, D.; SZEGEDY, C.; REED, S.; FU, C.-Y. and BERG, A. C.: “SSD: Single Shot MultiBox Detector,” in *Computer Vision – ECCV 2016*, Springer International Publishing, 2016, pp. 21–37. DOI: [10.1007/978-3-319-46448-0_2](https://doi.org/10.1007/978-3-319-46448-0_2) (cit. on p. 12).
- [20] REDMON, J.; DIVVALA, S.; GIRSHICK, R. and FARHADI, A.: “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. Accessed: Feb. 20, 2025. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Redmon_You_Only_Look_CVPR_2016_paper.html?ref=https://githubhelp.com (cit. on p. 12).
- [21] LAW, H. and DENG, J.: “CornerNet: Detecting Objects as Paired Keypoints,” presented at the Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 734–750. Accessed: Apr. 16, 2026. [Online]. Available: https://openaccess.thecvf.com/content_ECCV_2018/html/Hei_Law_CornerNet_Detecting_Objects_ECCV_2018_paper.html (cit. on p. 12).
- [22] ZHOU, X.; WANG, D. and KRÄHENBÜHL, P.: “Objects as Points.” arXiv: [1904.07850](https://arxiv.org/abs/1904.07850), Accessed: Apr. 16, 2026. [Online]. Available: <http://arxiv.org/abs/1904.07850> pre-published (cit. on pp. 12, 30).
- [23] TIAN, Z.; SHEN, C.; CHEN, H. and HE, T.: “FCOS: Fully Convolutional One-Stage Object Detection,” presented at the Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 9627–9636. Accessed: Apr. 16, 2026. [Online]. Available: https://openaccess.thecvf.com/content_ICCV_2019/html/Tian_FCOS_Fully_Convolutional_One-Stage_Object_Detection_ICCV_2019_paper.html (cit. on p. 12).
- [24] CARION, N.; MASSA, F.; SYNNAEVE, G.; USUNIER, N.; KIRILLOV, A. and ZAGORUYKO, S.: “End-to-End Object Detection with Transformers,” in *Computer Vision – ECCV 2020*, 2020, pp. 213–229. DOI: [10.1007/978-3-030-58452-8_13](https://doi.org/10.1007/978-3-030-58452-8_13) (cit. on p. 12).

- [25] VAN HORN, G.; MAC AODHA, O.; SONG, Y.; CUI, Y.; SUN, C.; SHEPARD, A.; ADAM, H.; PERONA, P. and BELONGIE, S.: “The iNaturalist Species Classification and Detection Dataset,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2018, pp. 8769–8778. DOI: [10.1109/CVPR.2018.00914](https://doi.org/10.1109/CVPR.2018.00914) Accessed: Feb. 9, 2025. [Online]. Available: <https://ieeexplore.ieee.org/document/8579012/> (cit. on p. 13).
- [26] WAH, C.; BRANSON, S.; WELINDER, P.; PERONA, P. and BELONGIE, S.: “The Caltech-UCSD Birds-200-2011 Dataset,” California Institute of Technology, CNS-TR-2011-001, 2011 (cit. on pp. 13, 25, 27, 33, 34, 36, 100–102, 105, 107).
- [27] EVERINGHAM, M.; VAN GOOL, L.; WILLIAMS, C. K. I.; WINN, J. and ZISSERMAN, A., “The Pascal Visual Object Classes (VOC) Challenge,” *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 1, 2010. DOI: [10.1007/s11263-009-0275-4](https://doi.org/10.1007/s11263-009-0275-4) Accessed: Feb. 20, 2025. [Online]. Available: <https://doi.org/10.1007/s11263-009-0275-4> (cit. on pp. 13, 29, 34).
- [28] LIN, T.-Y.; MAIRE, M.; BELONGIE, S.; HAYS, J.; PERONA, P.; RAMANAN, D.; DOLLÁR, P. and ZITNICK, C. L.: “Microsoft COCO: Common Objects in Context,” in *Computer Vision – ECCV 2014*, ser. Lecture Notes in Computer Science, Springer International Publishing, 2014, pp. 740–755. DOI: [10.1007/978-3-319-10602-1_48](https://doi.org/10.1007/978-3-319-10602-1_48) (cit. on pp. 13, 29, 33, 36, 46, 47, 52, 94, 98).
- [29] SHAO, S.; LI, Z.; ZHANG, T.; PENG, C.; YU, G.; ZHANG, X.; LI, J. and SUN, J.: “Objects365: A Large-Scale, High-Quality Dataset for Object Detection,” presented at the Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 8430–8439. Accessed: Jan. 9, 2026. [Online]. Available: https://openaccess.thecvf.com/content_ICCV_2019/html/Shao_Objects365_A_Large-Scale_High-Quality_Dataset_for_Object_Detection_ICCV_2019_paper.html (cit. on p. 13).
- [30] GUPTA, A.; DOLLAR, P. and GIRSHICK, R.: “LVIS: A Dataset for Large Vocabulary Instance Segmentation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5356–5364. Accessed: Feb. 20, 2025. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2019/html/Gupta_LVIS_A_Dataset_for_Large_Vocabulary_Instance_Segmentation_CVPR_2019_paper.html (cit. on pp. 13, 81).
- [31] DAVIS, J. and GOADRIC, M.: “The relationship between Precision-Recall and ROC curves,” in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06, Association for Computing Machinery, Jun. 25, 2006, pp. 233–240. DOI: [10.1145/1143844.1143874](https://doi.org/10.1145/1143844.1143874) Accessed: Feb. 20, 2026. [Online]. Available: <https://dl.acm.org/doi/10.1145/1143844.1143874> (cit. on p. 15).
- [32] DONAHUE, J.; JIA, Y.; VINYALS, O.; HOFFMAN, J.; ZHANG, N.; TZENG, E. and DARRELL, T.: “DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition,” in *Proceedings of the 31st International Conference on Machine Learning*, PMLR,

- Jan. 27, 2014, pp. 647–655. Accessed: Dec. 30, 2025. [Online]. Available: <https://proceedings.mlr.press/v32/donahue14.html> (cit. on p. 15).
- [33] YOSINSKI, J.; CLUNE, J.; BENGIO, Y. and LIPSON, H.: “How transferable are features in deep neural networks?” In *Advances in Neural Information Processing Systems*, vol. 27, Curran Associates, Inc., 2014. Accessed: Apr. 17, 2026. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2014/hash/532a2f85b6977104bc93f8580abbb330-Abstract.html (cit. on p. 16).
- [34] PAN, S. J. and YANG, Q., “A Survey on Transfer Learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010. DOI: [10.1109/TKDE.2009.191](https://doi.org/10.1109/TKDE.2009.191) Accessed: Apr. 17, 2026. [Online]. Available: <https://ieeexplore.ieee.org/document/5288526/> (cit. on p. 16).
- [35] ALAIN, G. and BENGIO, Y.: “Understanding intermediate layers using linear classifier probes,” presented at the ICLR 2017 Workshop, Feb. 17, 2017. Accessed: Jan. 6, 2026. [Online]. Available: <https://openreview.net/forum?id=HJ4-rAVtl> (cit. on p. 16).
- [36] CARON, M.; TOUVRON, H.; MISRA, I.; JÉGOU, H.; MAIRAL, J.; BOJANOWSKI, P. and JOULIN, A.: “Emerging Properties in Self-Supervised Vision Transformers,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9650–9660. Accessed: Dec. 29, 2023. [Online]. Available: https://openaccess.thecvf.com/content/ICCV2021/html/Caron_Emerging_Properties_in_Self-Supervised_Vision_Transformers_ICCV_2021_paper.html (cit. on pp. 16, 17, 55, 99, 101–103, 125).
- [37] AUSSERLECHNER, P.; HABERGER, D.; THALHAMMER, S.; WEIBEL, J.-B. and VINCZE, M.: “ZS6D: Zero-shot 6D Object Pose Estimation using Vision Transformers,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, May 2024, pp. 463–469. DOI: [10.1109/ICRA57147.2024.10611464](https://doi.org/10.1109/ICRA57147.2024.10611464) Accessed: May 1, 2026. [Online]. Available: <https://ieeexplore.ieee.org/document/10611464> (cit. on p. 16).
- [38] AMIR, S.; GANDELSMAN, Y.; BAGON, S. and DEKEL, T.: “On the Effectiveness of ViT Features as Local Semantic Descriptors,” in *Computer Vision – ECCV 2022 Workshops*, ser. Lecture Notes in Computer Science, Springer Nature Switzerland, 2023, pp. 39–55. DOI: [10.1007/978-3-031-25069-9_3](https://doi.org/10.1007/978-3-031-25069-9_3) [Online]. Available: <https://dino-vit-features.github.io/> (cit. on pp. 16, 53, 55, 99).
- [39] HADJIVELICHKOV, D.; ZWANE, S.; AGAPITO, L.; DEISENROTH, M. P. and KANOULAS, D.: “One-Shot Transfer of Affordance Regions? AffCorrs!,” presented at the 6th Annual Conference on Robot Learning, Aug. 15, 2022. Accessed: Jan. 3, 2024. [Online]. Available: <https://openreview.net/forum?id=GeM6VUwYinO> (cit. on p. 16).
- [40] OQUAB, M. et al., “DINOv2: Learning Robust Visual Features without Supervision,” *Transactions on Machine Learning Research*, Jul. 26, 2023. Accessed: Mar. 17, 2025. [Online]. Available: <https://openreview.net/forum?id=a68SUt6zFt> (cit. on pp. 16, 54, 55, 61, 63, 75, 80, 99, 101, 102, 109).

- [41] HINTON, G.; VINYALS, O. and DEAN, J.: “Distilling the Knowledge in a Neural Network,” in *NeurIPS 2014 Deep Learning Workshop*, Mar. 9, 2015. DOI: [10.48550/arXiv.1503.02531](https://doi.org/10.48550/arXiv.1503.02531) arXiv: [1503.02531](https://arxiv.org/abs/1503.02531) [stat]. Accessed: Apr. 18, 2026. [Online]. Available: <http://arxiv.org/abs/1503.02531> (cit. on p. 16).
- [42] DARCET, T.; OQUAB, M.; MAIRAL, J. and BOJANOWSKI, P.: “Vision Transformers Need Registers,” in *The Twelfth International Conference on Learning Representations*, Oct. 13, 2023. Accessed: Feb. 21, 2025. [Online]. Available: <https://openreview.net/forum?id=2dnO3LLiJ1> (cit. on p. 17).
- [43] RADFORD, A. et al.: “Learning Transferable Visual Models From Natural Language Supervision,” in *Proceedings of the 38th International Conference on Machine Learning*, PMLR, Jul. 1, 2021, pp. 8748–8763. Accessed: Feb. 3, 2025. [Online]. Available: <https://proceedings.mlr.press/v139/radford21a.html> (cit. on pp. 17, 20, 37).
- [44] LI, J.; LI, D.; SAVARESE, S. and HOI, S.: “BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models,” in *Proceedings of the 40th International Conference on Machine Learning*, PMLR, Jul. 3, 2023, pp. 19 730–19 742. Accessed: Mar. 10, 2026. [Online]. Available: <https://proceedings.mlr.press/v202/li23q.html> (cit. on pp. 17, 101, 102).
- [45] FANG, Y.; WANG, W.; XIE, B.; SUN, Q.; WU, L.; WANG, X.; HUANG, T.; WANG, X. and CAO, Y.: “EVA: Exploring the Limits of Masked Visual Representation Learning at Scale,” presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 19 358–19 369. Accessed: Apr. 18, 2026. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2023/html/Fang_EVA_Exploring_the_Limits_of_Masked_Visual_Representation_Learning_at_CVPR_2023_paper.html (cit. on pp. 17, 165).
- [46] ABNAR, S. and ZUIDEMA, W.: “Quantifying Attention Flow in Transformers,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Jul. 2020, pp. 4190–4197. DOI: [10.18653/v1/2020.acl-main.385](https://doi.org/10.18653/v1/2020.acl-main.385) Accessed: Jan. 19, 2026. [Online]. Available: <https://aclanthology.org/2020.acl-main.385/> (cit. on pp. 18, 55, 99).
- [47] CHEFER, H.; GUR, S. and WOLF, L.: “Transformer Interpretability Beyond Attention Visualization,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2021, pp. 782–791. DOI: [10.1109/CVPR46437.2021.00084](https://doi.org/10.1109/CVPR46437.2021.00084) Accessed: Dec. 1, 2024. [Online]. Available: <https://ieeexplore.ieee.org/document/9577970/> (cit. on pp. 19, 42, 44, 92).
- [48] BACH, S.; BINDER, A.; MONTAVON, G.; KLAUSCHEN, F.; MÜLLER, K.-R. and SAMEK, W., “On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation,” *PLOS ONE*, vol. 10, no. 7, e0130140, Jul. 10, 2015. DOI: [10.1371/journal.pone.0130140](https://doi.org/10.1371/journal.pone.0130140) Accessed: Apr. 19, 2026. [Online]. Available: <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0130140> (cit. on p. 19).

- [49] BINDER, A.; MONTAVON, G.; LAPUSCHKIN, S.; MÜLLER, K.-R. and SAMEK, W.: “Layer-Wise Relevance Propagation for Neural Networks with Local Renormalization Layers,” in *Artificial Neural Networks and Machine Learning – ICANN 2016*, Springer International Publishing, 2016, pp. 63–71. DOI: [10.1007/978-3-319-44781-0_8](https://doi.org/10.1007/978-3-319-44781-0_8) (cit. on p. 19).
- [50] KIM, B.; WATTENBERG, M.; GILMER, J.; CAI, C.; WEXLER, J.; VIEGAS, F. and SAYRES, R.: “Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV),” in *Proceedings of the 35th International Conference on Machine Learning*, PMLR, Jul. 3, 2018, pp. 2668–2677. Accessed: Feb. 4, 2024. [Online]. Available: <https://proceedings.mlr.press/v80/kim18d.html> (cit. on pp. 19, 35, 72, 75, 76, 103, 108).
- [51] WU, S.; YUKSEKGONUL, M.; ZHANG, L. and ZOU, J.: “Discover and Cure: Concept-aware Mitigation of Spurious Correlation,” in *Proceedings of the 40th International Conference on Machine Learning*, PMLR, Jul. 3, 2023, pp. 37 765–37 786. Accessed: Jan. 30, 2026. [Online]. Available: <https://proceedings.mlr.press/v202/wu23w.html> (cit. on pp. 20, 35, 72, 75, 76, 103, 125).
- [52] CHAO, W.-L.; CHANGPINYO, S.; GONG, B. and SHA, F.: “An Empirical Study and Analysis of Generalized Zero-Shot Learning for Object Recognition in the Wild,” in *Computer Vision – ECCV 2016*, Springer International Publishing, 2016, pp. 52–68. DOI: [10.1007/978-3-319-46475-6_4](https://doi.org/10.1007/978-3-319-46475-6_4) (cit. on p. 24).
- [53] FAN, Z.; MA, Y.; LI, Z. and SUN, J.: “Generalized Few-Shot Object Detection Without Forgetting,” presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 4527–4536. Accessed: Apr. 19, 2026. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2021/html/Fan_Generalized_Few-Shot_Object_Detection_Without_Forgetting_CVPR_2021_paper.html?ref=https://githubhelp.com (cit. on p. 25).
- [54] MAJI, S.; RAHTU, E.; KANNALA, J.; BLASCHKO, M. and VEDALDI, A. “Fine-Grained Visual Classification of Aircraft.” arXiv: [1306.5151](https://arxiv.org/abs/1306.5151) [cs], Accessed: Feb. 9, 2025. [Online]. Available: <http://arxiv.org/abs/1306.5151> pre-published (cit. on p. 25).
- [55] VAPNIK, V.: “Principles of Risk Minimization for Learning Theory,” in *Advances in Neural Information Processing Systems*, vol. 4, Morgan-Kaufmann, 1991. Accessed: Apr. 19, 2026. [Online]. Available: <https://proceedings.neurips.cc/paper/1991/hash/ff4d5fbbafdf976cfdc032e3bde78de5-Abstract.html> (cit. on p. 25).
- [56] D’AMOUR, A. et al., “Underspecification Presents Challenges for Credibility in Modern Machine Learning,” *Journal of Machine Learning Research*, vol. 23, no. 226, pp. 1–61, 2022. Accessed: May 26, 2026. [Online]. Available: <http://jmlr.org/papers/v23/20-1335.html> (cit. on p. 26).

- [57] TENEY, D.; PEYRARD, M. and ABBASNEJAD, E.: “Predicting Is Not Understanding: Recognizing and Addressing Underspecification in Machine Learning,” in *Computer Vision – ECCV 2022*, ser. Lecture Notes in Computer Science, Springer Nature Switzerland, 2022, pp. 458–476. DOI: [10.1007/978-3-031-20050-2_27](https://doi.org/10.1007/978-3-031-20050-2_27) (cit. on pp. 26, 71).
- [58] NEUHAUS, Y.; AUGUSTIN, M.; BOREIKO, V. and HEIN, M.: “Spurious Features Everywhere - Large-Scale Detection of Harmful Spurious Features in ImageNet,” in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, Oct. 1, 2023, pp. 20 178–20 189. DOI: [10.1109/ICCV51070.2023.01851](https://doi.org/10.1109/ICCV51070.2023.01851) Accessed: Feb. 8, 2026. [Online]. Available: <https://ieeexplore.ieee.org/document/10376625/> (cit. on pp. 26, 35).
- [59] RIBEIRO, M. T.; SINGH, S. and GUESTRIN, C.: “”Why Should I Trust You?”: Explaining the Predictions of Any Classifier,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’16, Association for Computing Machinery, Aug. 13, 2016, pp. 1135–1144. DOI: [10.1145/2939672.2939778](https://doi.org/10.1145/2939672.2939778) Accessed: May 5, 2026. [Online]. Available: <https://dl.acm.org/doi/10.1145/2939672.2939778> (cit. on p. 26).
- [60] SNELL, J.; SWERSKY, K. and ZEMEL, R.: “Prototypical Networks for Few-shot Learning,” in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017. Accessed: Jan. 30, 2024. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/hash/cb8da6767461f2812ae4290eac7cbc42-Abstract.html (cit. on pp. 26, 27, 29, 31).
- [61] KOCH, G.; ZEMEL, R. and SALAKHUTDINOV, R.: “Siamese Neural Networks for One-shot Image Recognition,” in *ICML Deep Learning Workshop*, Jul. 2015 (cit. on pp. 26, 28).
- [62] FINN, C.; ABBEEL, P. and LEVINE, S.: “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks,” in *Proceedings of the 34th International Conference on Machine Learning*, PMLR, Jul. 17, 2017, pp. 1126–1135. Accessed: Jan. 30, 2024. [Online]. Available: <https://proceedings.mlr.press/v70/finn17a.html> (cit. on pp. 26, 28).
- [63] VINYALS, O.; BLUNDELL, C.; LILLICRAP, T.; KAVUKCUOGLU, k. and WIERSTRA, D.: “Matching Networks for One Shot Learning,” in *Advances in Neural Information Processing Systems*, vol. 29, Curran Associates, Inc., 2016. Accessed: Jan. 30, 2024. [Online]. Available: <https://proceedings.neurips.cc/paper/2016/hash/90e1357833654983612fb05e3ec9148c-Abstract.html> (cit. on pp. 26–28, 32, 34).
- [64] FEI-FEI, L.; FERGUS, R. and PERONA, P.: “One-shot learning of object categories,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 594–611, Apr. 2006. DOI: [10.1109/TPAMI.2006.79](https://doi.org/10.1109/TPAMI.2006.79) Accessed: Apr. 20, 2026. [Online]. Available: <https://ieeexplore.ieee.org/document/1597116> (cit. on p. 26).

- [65] QIAO, Q.; GE, S.; WANG, K. and HU, H., “A Systematic Literature Review on Few-Shot Image Classification,” *IEEE Access*, vol. 13, pp. 181 154–181 179, 2025. DOI: [10.1109/ACCESS.2025.3622748](https://doi.org/10.1109/ACCESS.2025.3622748) Accessed: Apr. 20, 2026. [Online]. Available: <https://ieeexplore.ieee.org/document/11206340> (cit. on p. 27).
- [66] BANERJEE, A.; MERUGU, S.; DHILLON, I. and GHOSH, J.: “Clustering with Bregman Divergences,” in *Proceedings of the 2004 SIAM International Conference on Data Mining*, Society for Industrial and Applied Mathematics, Apr. 22, 2004, pp. 234–245. DOI: [10.1137/1.9781611972740.22](https://doi.org/10.1137/1.9781611972740.22) Accessed: Apr. 20, 2026. [Online]. Available: <https://epubs.siam.org/doi/10.1137/1.9781611972740.22> (cit. on p. 27).
- [67] ZHOU, F.; WANG, P.; ZHANG, L.; WEI, W. and ZHANG, Y.: “Revisiting Prototypical Network for Cross Domain Few-Shot Learning,” presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 20 061–20 070. Accessed: Dec. 15, 2023. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2023/html/Zhou_Revisiting_Prototypical_Network_for_Cross_Domain_Few-Shot_Learning_CVPR_2023_paper.html (cit. on pp. 27, 31, 34).
- [68] SHALAM, D. and KORMAN, S.: “The Balanced-Pairwise-Affinities Feature Transform,” in *Proceedings of the 41st International Conference on Machine Learning*, PMLR, Jul. 8, 2024, pp. 44 342–44 357. Accessed: Mar. 11, 2026. [Online]. Available: <https://proceedings.mlr.press/v235/shalam24a.html> (cit. on pp. 27, 31).
- [69] RAVI, S. and LAROCHELLE, H.: “Optimization as a Model for Few-Shot Learning,” presented at the International Conference on Learning Representations, Feb. 6, 2017. Accessed: Apr. 20, 2026. [Online]. Available: https://openreview.net/forum?id=rJY0-Kcll&ref=morioh.com&utm_source=morioh.com (cit. on pp. 28, 33).
- [70] KINGMA, D. P. and BA, J.: “Adam: A method for stochastic optimization,” in *International Conference on Learning Representations*, 2015. [Online]. Available: <https://arxiv.org/abs/1412.6980> (cit. on p. 28).
- [71] DUCHI, J.; HAZAN, E. and SINGER, Y., “Adaptive subgradient methods for online learning and stochastic optimization,” *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011 (cit. on p. 28).
- [72] LAKE, B. M.; SALAKHUTDINOV, R.; GROSS, J. and TENENBAUM, J. B.: “One shot learning of simple visual concepts,” in *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 33, 2011 (cit. on pp. 28, 32).
- [73] HELD, D.; THRUN, S. and SAVARESE, S.: “Robust single-view instance recognition,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, May 2016, pp. 2152–2159. DOI: [10.1109/ICRA.2016.7487365](https://doi.org/10.1109/ICRA.2016.7487365) Accessed: Dec. 29, 2023. [Online]. Available: <http://ieeexplore.ieee.org/document/7487365/> (cit. on p. 28).

- [74] SINGH, A.; SHA, J.; NARAYAN, K. S.; ACHIM, T. and ABBEEL, P.: “BigBIRD: A large-scale 3D database of object instances,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 509–516. DOI: [10.1109/ICRA.2014.6906903](https://doi.org/10.1109/ICRA.2014.6906903) Accessed: Aug. 1, 2024. [Online]. Available: <https://ieeexplore.ieee.org/document/6906903/?arnumber=6906903> (cit. on p. 28).
- [75] LAI, K.; BO, L.; REN, X. and FOX, D.: “A large-scale hierarchical multi-view RGB-D object dataset,” in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 1817–1824. DOI: [10.1109/ICRA.2011.5980382](https://doi.org/10.1109/ICRA.2011.5980382) Accessed: Aug. 1, 2024. [Online]. Available: <https://ieeexplore.ieee.org/document/5980382/?arnumber=5980382> (cit. on p. 28).
- [76] KÖHLER, M.; EISENBACH, M. and GROSS, H.-M., “Few-Shot Object Detection: A Comprehensive Survey,” *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2023. DOI: [10.1109/TNNLS.2023.3265051](https://doi.org/10.1109/TNNLS.2023.3265051) Accessed: Jun. 23, 2024. [Online]. Available: <https://ieeexplore.ieee.org/document/10103630> (cit. on p. 29).
- [77] XIN, Z.; CHEN, S.; WU, T.; SHAO, Y.; DING, W. and YOU, X., “Few-shot object detection: Research advances and challenges,” *Information Fusion*, vol. 107, p. 102 307, Jul. 2024. DOI: [10.1016/j.inffus.2024.102307](https://doi.org/10.1016/j.inffus.2024.102307) Accessed: Mar. 2, 2025. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S156625352400085X> (cit. on p. 29).
- [78] LIU, T.; ZHANG, L.; WANG, Y.; GUAN, J.; FU, Y.; ZHAO, J. and ZHOU, S., “Recent Few-shot Object Detection Algorithms: A Survey with Performance Comparison,” *ACM Transactions on Intelligent Systems and Technology*, vol. 14, no. 4, pp. 1–36, Aug. 31, 2023. DOI: [10.1145/3593588](https://doi.org/10.1145/3593588) Accessed: Jul. 30, 2024. [Online]. Available: <https://dl.acm.org/doi/10.1145/3593588> (cit. on p. 29).
- [79] WANG, Y.; YAO, Q.; KWOK, J. T. and NI, L. M., “Generalizing from a Few Examples: A Survey on Few-shot Learning,” *ACM Comput. Surv.*, vol. 53, no. 3, pp. 63:1–63:34, Jun. 12, 2020. DOI: [10.1145/3386252](https://doi.org/10.1145/3386252) Accessed: Apr. 20, 2026. [Online]. Available: <https://dl.acm.org/doi/10.1145/3386252> (cit. on p. 29).
- [80] KANG, B.; LIU, Z.; WANG, X.; YU, F.; FENG, J. and DARRELL, T.: “Few-Shot Object Detection via Feature Reweighting,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, Oct. 2019, pp. 8419–8428. DOI: [10.1109/ICCV.2019.00851](https://doi.org/10.1109/ICCV.2019.00851) Accessed: Feb. 19, 2025. [Online]. Available: <https://ieeexplore.ieee.org/document/9010944/> (cit. on pp. 29, 30, 34, 60).
- [81] YAN, X.; CHEN, Z.; XU, A.; WANG, X.; LIANG, X. and LIN, L.: “Meta R-CNN: Towards General Solver for Instance-Level Low-Shot Learning,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, Oct. 2019, pp. 9576–9585. DOI: [10.1109/ICCV.2019.00967](https://doi.org/10.1109/ICCV.2019.00967) Accessed: Feb. 10, 2025. [Online]. Available: <https://ieeexplore.ieee.org/document/9011009/> (cit. on pp. 29, 30, 60).
- [82] HE, K.; GKIOXARI, G.; DOLLÁR, P. and GIRSHICK, R.: “Mask R-CNN,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 2980–2988. DOI: [10.1109/ICCV.2017.322](https://doi.org/10.1109/ICCV.2017.322) (cit. on pp. 29, 31, 47, 164).

- [83] QIAO, L.; ZHAO, Y.; LI, Z.; QIU, X.; WU, J. and ZHANG, C.: “DeFRCN: Decoupled Faster R-CNN for Few-Shot Object Detection,” presented at the Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 8681–8690. Accessed: Apr. 21, 2024. [Online]. Available: https://openaccess.thecvf.com/content/ICCV2021/html/Qiao_DeFRCN_Decoupled_Faster_R-CNN_for_Few-Shot_Object_Detection_ICCV_2021_paper.html (cit. on p. 29).
- [84] WANG, X.; HUANG, T.; GONZALEZ, J.; DARRELL, T. and YU, F.: “Frustratingly Simple Few-Shot Object Detection,” in *Proceedings of the 37th International Conference on Machine Learning*, PMLR, Nov. 21, 2020, pp. 9919–9928. Accessed: Apr. 21, 2024. [Online]. Available: <https://proceedings.mlr.press/v119/wang20j.html> (cit. on pp. 30, 34).
- [85] LI, B.; WANG, C.; REDDY, P.; KIM, S. and SCHERER, S.: “AirDet: Few-Shot Detection Without Fine-Tuning for Autonomous Exploration,” in *Computer Vision – ECCV 2022*, Springer Nature Switzerland, 2022, pp. 427–444. DOI: [10.1007/978-3-031-19842-7_25](https://doi.org/10.1007/978-3-031-19842-7_25) (cit. on pp. 30, 60).
- [86] BULAT, A.; GUERRERO, R.; MARTINEZ, B. and TZIMIROPOULOS, G.: “FS-DETR: Few-Shot DETection TRansformer with Prompting and without Re-Training,” presented at the Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 11 793–11 802. Accessed: Feb. 22, 2026. [Online]. Available: https://openaccess.thecvf.com/content/ICCV2023/html/Bulat_FS-DETR_Few-Shot_DETetection_TRansformer_with_Prompting_and_without_Re-Training_ICCV_2023_paper.html (cit. on pp. 30, 60).
- [87] WANG, Z.; LI, B.; WANG, C. and SCHERER, S.: “AirShot: Efficient Few-Shot Detection for Autonomous Exploration,” in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Oct. 14, 2024, pp. 11 654–11 661. DOI: [10.1109/IROS58592.2024.10801738](https://doi.org/10.1109/IROS58592.2024.10801738) Accessed: May 4, 2026. [Online]. Available: <https://ieeexplore.ieee.org/document/10801738/> (cit. on pp. 30, 60).
- [88] ZHANG, X.; LIU, Y.; WANG, Y. and BOULARIAS, A.: “Detect Everything with Few Examples,” in *8th Annual Conference on Robot Learning (CoRL)*, Sep. 5, 2024. Accessed: Jan. 31, 2025. [Online]. Available: <https://openreview.net/forum?id=HlxRd529nG> (cit. on pp. 30, 31, 60, 62, 63, 65, 78, 80, 82, 109–113, 125).
- [89] PEREZ-RUA, J.-M.; ZHU, X.; HOSPEDALES, T. M. and XIANG, T.: “Incremental Few-Shot Object Detection,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2020, pp. 13 843–13 852. DOI: [10.1109/CVPR42600.2020.01386](https://doi.org/10.1109/CVPR42600.2020.01386) Accessed: Jun. 23, 2024. [Online]. Available: <https://ieeexplore.ieee.org/document/9157715/> (cit. on p. 30).
- [90] NGUYEN, K. and TODOROVIC, S.: “iFS-RCNN: An Incremental Few-Shot Instance Seg-
menter,” presented at the Proceedings of the IEEE/CVF Conference on Computer Vi-
sion and Pattern Recognition, 2022, pp. 7010–7019. Accessed: Jun. 23, 2024. [Online].
Available: https://openaccess.thecvf.com/content/CVPR2022/html/Nguyen_iFS-

- [RCNN_An_Incremental_Few-Shot_Instance_Segmenter_CVPR_2022_paper.html](#) (cit. on p. 31).
- [91] ZHANG, L.; WENG, L.; ZHANG, Y. and GAO, F., “Incremental few-shot instance segmentation without fine-tuning on novel classes,” *Computer Vision and Image Understanding*, vol. 254, p. 104 323, Mar. 1, 2025. DOI: [10.1016/j.cviu.2025.104323](#) Accessed: Apr. 20, 2026. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1077314225000463> (cit. on p. 31).
- [92] GANEA, D. A.; BOOM, B. and POPPE, R.: “Incremental Few-Shot Instance Segmentation,” presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 1185–1194. Accessed: Jun. 23, 2024. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2021/html/Ganea_Incremental_Few-Shot_Instance_Segmentation_CVPR_2021_paper.html (cit. on p. 31).
- [93] REN, M.; TRIANTAFILLOU, E.; RAVI, S.; SNELL, J.; SWERSKY, K.; TENENBAUM, J. B.; LAROCHELLE, H. and ZEMEL, R. S.: “Meta-Learning for Semi-Supervised Few-Shot Classification,” presented at the International Conference on Learning Representations, Feb. 15, 2018. Accessed: Apr. 24, 2026. [Online]. Available: <https://openreview.net/forum?id=HJcSzz-CZ> (cit. on p. 33).
- [94] BERTINETTO, L.; HENRIQUES, J. F.; TORR, P. and VEDALDI, A.: “Meta-learning with differentiable closed-form solvers,” presented at the International Conference on Learning Representations, 2019. Accessed: Apr. 21, 2026. [Online]. Available: <https://openreview.net/forum?id=HyxnZh0ct7> (cit. on p. 33).
- [95] MANGLA, P.; KUMARI, N.; SINHA, A.; SINGH, M.; KRISHNAMURTHY, B. and BALASUBRAMANIAN, V. N.: “Charting the Right Manifold: Manifold Mixup for Few-shot Learning,” presented at the Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2020, pp. 2218–2227. Accessed: Apr. 24, 2026. [Online]. Available: https://openaccess.thecvf.com/content_WACV_2020/html/Mangla_Charting_the_Right_Manifold_Manifold_Mixup_for_Few-shot_Learning_WACV_2020_paper.html (cit. on p. 33).
- [96] HU, Y.; GRIPON, V. and PATEUX, S.: “Leveraging the Feature Distribution in Transfer-Based Few-Shot Learning,” in *Artificial Neural Networks and Machine Learning – ICANN 2021*, Springer International Publishing, 2021, pp. 487–499. DOI: [10.1007/978-3-030-86340-1_39](#) (cit. on p. 33).
- [97] CHEN, X. and WANG, G.: “Few-Shot Learning by Integrating Spatial and Frequency Representation,” in *2021 18th Conference on Robots and Vision (CRV)*, May 2021, pp. 49–56. DOI: [10.1109/CRV52889.2021.00011](#) Accessed: Apr. 2, 2026. [Online]. Available: <https://ieeexplore.ieee.org/document/9469423> (cit. on p. 33).

- [98] BANSAL, A.; SIKKA, K.; SHARMA, G.; CHELLAPPA, R. and DIVAKARAN, A.: “Zero-Shot Object Detection,” in *Computer Vision – ECCV 2018*, vol. 11205, Springer International Publishing, 2018, pp. 397–414. DOI: [10.1007/978-3-030-01246-5_24](https://doi.org/10.1007/978-3-030-01246-5_24) Accessed: Dec. 30, 2023. [Online]. Available: https://link.springer.com/10.1007/978-3-030-01246-5_24 (cit. on p. 33).
- [99] ZAREIAN, A.; ROSA, K. D.; HU, D. H. and CHANG, S.-F.: “Open-Vocabulary Object Detection Using Captions,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2021, pp. 14 388–14 397. DOI: [10.1109/CVPR46437.2021.01416](https://doi.org/10.1109/CVPR46437.2021.01416) Accessed: Dec. 30, 2023. [Online]. Available: <https://ieeexplore.ieee.org/document/9577418/> (cit. on p. 34).
- [100] XIONG, W.: “CD-FSOD: A Benchmark For Cross-Domain Few-Shot Object Detection,” in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Jun. 2023, pp. 1–5. DOI: [10.1109/ICASSP49357.2023.10096216](https://doi.org/10.1109/ICASSP49357.2023.10096216) Accessed: Jun. 27, 2025. [Online]. Available: <https://ieeexplore.ieee.org/document/10096216/> (cit. on pp. 34, 128).
- [101] BUEHL, D.: *Classroom Strategies for Interactive Learning*, 4th ed. Routledge, Oct. 10, 2023, 270 pp. DOI: [10.4324/9781032680842](https://doi.org/10.4324/9781032680842) (cit. on p. 35).
- [102] BOHG, J.; HAUSMAN, K.; SANKARAN, B.; BROCK, O.; KRAGIC, D.; SCHAAAL, S. and SUKHATME, G. S.: “Interactive Perception: Leveraging Action in Perception and Perception in Action,” *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1273–1291, Dec. 2017. DOI: [10.1109/TRO.2017.2721939](https://doi.org/10.1109/TRO.2017.2721939) Accessed: Apr. 25, 2026. [Online]. Available: <https://ieeexplore.ieee.org/document/8007233> (cit. on p. 35).
- [103] SETTLES, B.: “Active Learning Literature Survey,” University of Wisconsin-Madison Department of Computer Sciences, 2009. [Online]. Available: <https://burrsettles.com/pub/settles.activelearning.pdf> (cit. on p. 35).
- [104] WAH, C.; BRANSON, S.; PERONA, P. and BELONGIE, S.: “Multiclass recognition and part localization with humans in the loop,” in *2011 International Conference on Computer Vision*, Nov. 2011, pp. 2524–2531. DOI: [10.1109/ICCV.2011.6126539](https://doi.org/10.1109/ICCV.2011.6126539) Accessed: Feb. 14, 2025. [Online]. Available: <https://ieeexplore.ieee.org/document/6126539/?arnumber=6126539> (cit. on p. 36).
- [105] CHEN, C.; LI, O.; TAO, D.; BARNETT, A.; RUDIN, C. and SU, J. K.: “This Looks Like That: Deep Learning for Interpretable Image Recognition,” in *Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc., 2019. Accessed: Jan. 30, 2024. [Online]. Available: <https://proceedings.neurips.cc/paper/2019/hash/adf7ee2dcf142b0e11888e72b43fcb75-Abstract.html> (cit. on pp. 36, 53, 57, 60).
- [106] BONTEMPELLI, A.; TESO, S.; TENTORI, K.; GIUNCHIGLIA, F. and PASSERINI, A.: “Concept-level Debugging of Part-Prototype Networks,” in *The Eleventh International Conference on Learning Representations*, Sep. 29, 2022. Accessed: Jan. 29, 2024. [Online]. Available: <https://openreview.net/forum?id=oiwXWPDyNk> (cit. on pp. 36, 53, 115).

- [107] XUE, M.; HUANG, Q.; ZHANG, H.; HU, J.; SONG, J.; SONG, M. and JIN, C.: “ProtoP-Former: Concentrating on Prototypical Parts in Vision Transformers for Interpretable Image Recognition,” presented at the Thirty-Third International Joint Conference on Artificial Intelligence, vol. 2, Aug. 1, 2024, pp. 1516–1524. DOI: [10.24963/ijcai.2024/168](https://doi.org/10.24963/ijcai.2024/168) Accessed: Jan. 19, 2026. [Online]. Available: <https://www.ijcai.org/proceedings/2024/168> (cit. on pp. 36, 53, 55, 60, 102).
- [108] DEHGHAN, M.; ZHANG, Z.; SIAM, M.; JIN, J.; PETRICH, L. and JAGERSAND, M.: “Online Object and Task Learning via Human Robot Interaction,” in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 2132–2138. DOI: [10.1109/ICRA.2019.8794036](https://doi.org/10.1109/ICRA.2019.8794036) (cit. on p. 36).
- [109] ZHOU, B.; KHOSLA, A.; LAPEDRIZA, A.; OLIVA, A. and TORRALBA, A.: “Learning Deep Features for Discriminative Localization,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 2921–2929. DOI: [10.1109/CVPR.2016.319](https://doi.org/10.1109/CVPR.2016.319) Accessed: Oct. 21, 2024. [Online]. Available: <https://ieeexplore.ieee.org/document/7780688/> (cit. on p. 36).
- [110] COVER, T. and HART, P., “Nearest neighbor pattern classification,” *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967. DOI: [10.1109/TIT.1967.1053964](https://doi.org/10.1109/TIT.1967.1053964) Accessed: Oct. 21, 2024. [Online]. Available: <https://ieeexplore.ieee.org/document/1053964/?arnumber=1053964> (cit. on p. 36).
- [111] BERG, S. et al., “Ilastik: Interactive machine learning for (bio)image analysis,” *Nature Methods*, vol. 16, no. 12, pp. 1226–1232, Dec. 2019. DOI: [10.1038/s41592-019-0582-9](https://doi.org/10.1038/s41592-019-0582-9) Accessed: Apr. 25, 2026. [Online]. Available: <https://www.nature.com/articles/s41592-019-0582-9> (cit. on p. 37).
- [112] BREIMAN, L., “Random Forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 1, 2001. DOI: [10.1023/A:1010933404324](https://doi.org/10.1023/A:1010933404324) Accessed: May 5, 2026. [Online]. Available: <https://doi.org/10.1023/A:1010933404324> (cit. on p. 37).
- [113] BOHUS, D.; ANDRIST, S.; FENIELLO, A.; SAW, N. and HORVITZ, E.: “Continual Learning about Objects in the Wild: An Interactive Approach,” in *Proceedings of the 2022 International Conference on Multimodal Interaction*, ser. ICMI ’22, Association for Computing Machinery, Nov. 7, 2022, pp. 476–486. DOI: [10.1145/3536221.3556567](https://doi.org/10.1145/3536221.3556567) Accessed: Jan. 8, 2026. [Online]. Available: <https://doi.org/10.1145/3536221.3556567> (cit. on p. 37).
- [114] LI, X.; DING, H.; YUAN, H.; ZHANG, W.; PANG, J.; CHENG, G.; CHEN, K.; LIU, Z. and LOY, C. C., “Transformer-Based Visual Segmentation: A Survey,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 12, pp. 10 138–10 163, Dec. 2024. DOI: [10.1109/TPAMI.2024.3434373](https://doi.org/10.1109/TPAMI.2024.3434373) Accessed: Dec. 2, 2024. [Online]. Available: <https://ieeexplore.ieee.org/document/10613466/?arnumber=10613466> (cit. on p. 41).

- [115] ZHANG, B.; TIAN, Z.; TANG, Q.; CHU, X.; WEI, X.; SHEN, C. and LIU, Y., “SegViT: Semantic Segmentation with Plain Vision Transformers,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 4971–4982, Dec. 6, 2022. Accessed: Dec. 2, 2024. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/hash/20189b1aaa8eddb6d8bd6c1067ab5f3f-Abstract-Conference.html (cit. on p. 41).
- [116] BIANCO, S.; GAVIRAGHI, E. and SCHETTINI, R.: “Efficient Deep Learning Models for Litter Detection in the Wild,” in *2024 IEEE 8th Forum on Research and Technologies for Society and Industry Innovation (RTSI)*, Sep. 2024, pp. 601–606. DOI: [10.1109/RTSI61910.2024.10761805](https://doi.org/10.1109/RTSI61910.2024.10761805) Accessed: Nov. 29, 2024. [Online]. Available: <https://ieeexplore.ieee.org/document/10761805/> (cit. on p. 41).
- [117] PROENÇA, P. F. and SIMÕES, P. “TACO: Trash Annotations in Context for Litter Detection.” arXiv: [2003.06975](https://arxiv.org/abs/2003.06975), Accessed: Nov. 29, 2024. [Online]. Available: <http://arxiv.org/abs/2003.06975> pre-published (cit. on pp. 44, 86–88).
- [118] DWIBEDI, D.; MISRA, I. and HEBERT, M.: “Cut, Paste and Learn: Surprisingly Easy Synthesis for Instance Detection,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, IEEE, Oct. 2017, pp. 1310–1319. DOI: [10.1109/ICCV.2017.146](https://doi.org/10.1109/ICCV.2017.146) Accessed: Mar. 17, 2023. [Online]. Available: <http://ieeexplore.ieee.org/document/8237408/> (cit. on pp. 46, 50).
- [119] DJUKIC, N.; KROPATSCH, W. and VINCZE, M.: “The Difficulties of Detecting Deformable Objects Using Deep Neural Networks,” presented at the Proceedings of the Joint Austrian Computer Vision and Robotics Workshop 2020, 2020, p. 6. DOI: [10.3217/978-3-85125-752-6-30](https://doi.org/10.3217/978-3-85125-752-6-30) Accessed: Apr. 30, 2026. [Online]. Available: <https://repositum.tuwien.at/handle/20.500.12708/55558> (cit. on pp. 46, 50).
- [120] GHIASI, G.; CUI, Y.; SRINIVAS, A.; QIAN, R.; LIN, T.-Y.; CUBUK, E. D.; LE, Q. V. and ZOPH, B.: “Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2021, pp. 2917–2927. DOI: [10.1109/CVPR46437.2021.00294](https://doi.org/10.1109/CVPR46437.2021.00294) Accessed: Mar. 20, 2025. [Online]. Available: <https://ieeexplore.ieee.org/document/9578639/> (cit. on pp. 46, 50).
- [121] BORMANN, R.; WANG, X.; XU, J. and SCHMIDT, J.: “DirtNet: Visual Dirt Detection for Autonomous Cleaning Robots,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 1977–1983. DOI: [10.1109/ICRA40945.2020.9196559](https://doi.org/10.1109/ICRA40945.2020.9196559) Accessed: Jan. 18, 2026. [Online]. Available: <https://ieeexplore.ieee.org/document/9196559> (cit. on p. 50).
- [122] HÖNIG, P.; THALHAMMER, S.; WEIBEL, J.-B.; HIRSCHMANNER, M. and VINCZE, M.: “Shape-Biased Texture Agnostic Representations for Improved Textureless and Metallic Object Detection and 6D Pose Estimation,” in *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Feb. 2025, pp. 8806–8815. DOI: [10.1109/WACV61041.2025.00853](https://doi.org/10.1109/WACV61041.2025.00853) Accessed: Apr. 30, 2026. [Online]. Available: <https://ieeexplore.ieee.org/document/10943969> (cit. on p. 51).

- [123] OLAH, C.; MORDVINTSEV, A. and SCHUBERT, L., “Feature visualization,” *Distill*, vol. 2, no. 11, e7, 2017 (cit. on p. 53).
- [124] BAU, D.; ZHU, J.-Y.; STROBELT, H.; LAPEDRIZA, A.; ZHOU, B. and TORRALBA, A., “Understanding the role of individual units in a deep neural network,” *Proceedings of the National Academy of Sciences*, vol. 117, no. 48, pp. 30 071–30 078, Dec. 2020. DOI: [10.1073/pnas.1907375117](https://doi.org/10.1073/pnas.1907375117) Accessed: Mar. 7, 2023. [Online]. Available: <https://www.pnas.org/doi/10.1073/pnas.1907375117> (cit. on p. 53).
- [125] EISENBACH, M.; FRANKE, H.; FRANZE, E.; KÖHLER, M.; AGANIAN, D.; SEICHTER, D. and GROSS, H.-M., “Detection of Novel Objects without Fine-Tuning in Assembly Scenarios by Class-Agnostic Object Detection and Object Re-Identification,” *Automation*, vol. 5, no. 3, pp. 373–406, 3 Sep. 2024. DOI: [10.3390/automation5030023](https://doi.org/10.3390/automation5030023) Accessed: Mar. 2, 2025. [Online]. Available: <https://www.mdpi.com/2673-4052/5/3/23> (cit. on p. 60).
- [126] KAUFMAN, L. and ROUSSEEUW, P. J.: “Clustering by means of medoids,” in *Proceedings of the Statistical Data Analysis Based on the L1 Norm Conference*, (Neuchâtel, Switzerland), 1987, pp. 405–416 (cit. on pp. 64, 115).
- [127] ZHANG, X.: *mlzxy/devit*, Feb. 13, 2026. Accessed: Feb. 15, 2026. [Online]. Available: <https://github.com/mlzxy/devit> (cit. on pp. 66, 168).
- [128] BAU, D.; ZHOU, B.; KHOSLA, A.; OLIVA, A. and TORRALBA, A.: “Network Dissection: Quantifying Interpretability of Deep Visual Representations,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jul. 2017, pp. 3319–3327. DOI: [10.1109/CVPR.2017.354](https://doi.org/10.1109/CVPR.2017.354) Accessed: Jan. 27, 2024. [Online]. Available: <http://ieeexplore.ieee.org/document/8099837/> (cit. on pp. 75, 76, 103).
- [129] KIRILLOV, A. et al.: “Segment Anything,” presented at the Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 4015–4026. Accessed: Jan. 14, 2026. [Online]. Available: https://openaccess.thecvf.com/content/ICCV2023/html/Kirillov_SegmentAnything_ICCV_2023_paper.html (cit. on pp. 79, 80).
- [130] CARION, N. et al.: “SAM 3: Segment Anything with Concepts,” presented at the The Fourteenth International Conference on Learning Representations, Oct. 8, 2025. Accessed: May 4, 2026. [Online]. Available: <https://openreview.net/forum?id=r35clVtGzw> (cit. on pp. 80, 129).
- [131] TOUVRON, H.; CORD, M.; EL-NOUBY, A.; VERBEEK, J. and JÉGOU, H.: “Three Things Everyone Should Know About Vision Transformers,” in *Computer Vision – ECCV 2022*, Springer Nature Switzerland, 2022, pp. 497–515. DOI: [10.1007/978-3-031-20053-3_29](https://doi.org/10.1007/978-3-031-20053-3_29) (cit. on p. 86).
- [132] SHALAM, D. and KORMAN, S.: “The Self-Optimal-Transport Feature Transform.” arXiv: [2204.03065](https://arxiv.org/abs/2204.03065) [cs], Accessed: Feb. 5, 2024. [Online]. Available: <http://arxiv.org/abs/2204.03065> pre-published (cit. on p. 102).

-
- [133] GRIGOREV, A. “Alexeygrigorev/clothing-dataset-small,” Accessed: Mar. 2, 2026. [Online]. Available: <https://github.com/alexeygrigorev/clothing-dataset-small> (cit. on pp. 103, 105).
- [134] CALLI, B.; WALSMAN, A.; SINGH, A.; SRINIVASA, S.; ABBEEL, P. and DOLLAR, A. M., “Benchmarking in Manipulation Research: Using the Yale-CMU-Berkeley Object and Model Set,” *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 36–52, Sep. 2015. DOI: [10.1109/MRA.2015.2448951](https://doi.org/10.1109/MRA.2015.2448951) Accessed: Jan. 28, 2024. [Online]. Available: <https://ieeexplore.ieee.org/document/7254318> (cit. on p. 106).
- [135] HUME, G. “English: Cordless drill,” Accessed: Jan. 28, 2024. [Online]. Available: <https://commons.wikimedia.org/wiki/File:CordlessDrill.jpg> (cit. on p. 106).
- [136] JUNG, M. J.; HAN, S. D. and KIM, J., “Re-scoring using image-language similarity for few-shot object detection,” *Computer Vision and Image Understanding*, vol. 241, p. 103 956, Apr. 2024. DOI: [10.1016/j.cviu.2024.103956](https://doi.org/10.1016/j.cviu.2024.103956) Accessed: Feb. 21, 2025. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S1077314224000377> (cit. on p. 113).
- [137] SIMÉONI, O. et al. “DINOv3.” arXiv: [2508.10104](https://arxiv.org/abs/2508.10104) [cs], Accessed: Jan. 31, 2026. [Online]. Available: <http://arxiv.org/abs/2508.10104> pre-published (cit. on p. 128).
- [138] BOLYA, D. et al.: “Perception Encoder: The best visual embeddings are not at the output of the network,” presented at the The Thirty-ninth Annual Conference on Neural Information Processing Systems, Oct. 29, 2025. Accessed: May 4, 2026. [Online]. Available: <https://openreview.net/forum?id=INqB0mwIpG> (cit. on p. 128).
- [139] ESPINOSA, M.; YANG, C.; ERICSSON, L.; McDONAGH, S. and CROWLEY, E. J. “No time to train! Training-Free Reference-Based Instance Segmentation.” arXiv: [2507.02798](https://arxiv.org/abs/2507.02798), Accessed: May 6, 2026. [Online]. Available: <http://arxiv.org/abs/2507.02798> pre-published (cit. on p. 128).

List of Figures

1.1	General concept overview	5
4.1	Concept overview for pretraining-based few-shot perception	39
4.2	Model architecture for litter segmentation	42
4.3	Hybrid inference pipeline with depth fusion	48
4.4	Instance-based dataset augmentation	51
4.5	VINTER model	56
4.6	DE-fine-ViT model	62
5.1	Concept overview for interaction-based few-shot perception	71
5.2	Procedure for interactively adjusting concept sensitivity	74
5.3	Prototype modification using CAVs	77
5.4	Interactive DE-fine-ViT model	82
6.1	Qualitative TACO segmentation example	87
6.2	Binary ViT-B layer-specific segmentation maps	88
6.3	Layer-specific segmentation performances on the TACO dataset	88
6.4	Litter segmentation of Mask R-CNN and ViT-B	91
6.5	Qualitative examples for unseen waste items	91
6.6	Qualitative comparison of different attention masking methods	92
6.7	Sensor inputs used for sorting batteries	95
6.8	Processing steps of the VINTER model	100
6.9	Qualitative results for CAV-based sorting of the small clothing dataset	105
6.10	CUB-200-2011 examples for concept associations	105
6.11	Interactive use case for the VINTER model	106
6.12	Qualitative DE-ViT limitations	110
6.13	Mini-tool-dataset	111
6.14	Visualization of the part-based prediction refinement	111
6.15	Categories of the fine-grained power drill dataset	112
6.16	Similarity matrices for class and part prototypes	114
6.17	Analysis of multi-layer part prototypes for DE-fine-ViT	116
6.18	ROBDEKON scenario for hazard retrieval	118
6.19	Detection of a hazardous container	119
6.20	Components of the battery sorting demonstrator	119
6.21	i-SAMLR web UI	120
A.1	Two samples from the fine-grained power drill dataset	169

List of Tables

5.1	Comparison of the non-interactive and interactive VINTER modes	72
5.2	Comparison of the non-interactive and interactive DE-fine-ViT modes	83
6.1	Pixel-level TACO performance for different Gaussian filters	89
6.2	Segmentation performance for different head fusion methods	90
6.3	Segmentation performance of different attention masking methods	93
6.4	Collection of batteries	94
6.5	Battery classification performance on the test dataset	96
6.6	Set of geometric verification rules	97
6.7	Comparison of different backbones on CUB-200-2011	102
6.8	Comparison with state-of-the-art methods on the CUB-200-2011	102
6.9	Color concept prediction across facets and backbones	104
6.10	Performance evaluation (mAP) on the mini-tool-dataset	111
6.11	Performance evaluation (TP, FP, FN) on the mini-tool-dataset	112
6.12	Parts of the fine-grained power drill dataset	112
6.13	Performance evaluation (mAP) on the fine-grained power drill dataset	113
6.14	Precision and recall on the fine-grained power drill dataset	113
6.15	Best-case per-class performance on the fine-grained power drill dataset	116
A.1	Mask R-CNN training configuration for LiViT	164
A.2	Mask R-CNN training configuration	164
A.3	Implementation details of VINTER backbones	165
A.4	Implementation details of the VINTER modules	166
A.5	Texture concept prediction across facets and backbones	167
A.6	Default DE-fine-ViT configuration	168

Acronyms

AP	Average Precision
AUROC	Area Under the Receiver Operating Characteristic curve
BLIP-2	Bootstrapping Language-Image Pre-training v2
CAV	Concept Activation Vector
CLIP	Contrastive Language-Image Pre-training
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DE-fine-ViT	Detect Everything fine-grained Vision Transformer
DETR	DEtection TRansformer
DE-ViT	Detect Everything Vision Transformer
DINO	self-DIstillation with NO labels
DISC	Discover and Cure
Fast R-CNN	Fast R-CNN
Faster R-CNN	Faster R-CNN
FCOS	Fully Convolutional One-Stage object detector
FLOPS	Floating-Point Operations Per Second
FN	False Negative
FP	False Positive
FPR	False Positive Rate
FRM	Fine-grained Recognition Module
FSIC	Few-Shot Image Classification
FSL	Few-Shot Learning
FSOD	Few-Shot Object Detection

FSOR	Few-Shot Object Recognition
GPU	Graphics Processing Unit
IFM	Information Fusion Module
iFSIS	incremental Few-Shot Instance Segmentation
iFSOD	incremental Few-Shot Object Detection
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
IoU	Intersection over Union
LRP	Layer-wise Relevance Propagation
LSTM	Long Short-Term Memory
MAML	Model-Agnostic Meta-Learning
mAP	mean Average Precision
Mask R-CNN	Mask R-CNN
Meta R-CNN	Meta R-CNN
NMS	Non-Maximum Suppression
OOD	Out-of-Distribution
R-CNN	Region-based Convolutional Neural Network
ResNet	Residual Network
RGB	Red, Green, and Blue
RGB-D	Red, Green, Blue, and Depth
ROC	Receiver Operating Characteristic
RoI	Region of Interest
RPN	Region Proposal Network
SAM	Segment Anything Model
SSD	Single Shot MultiBox Detector
SVM	Support Vector Machine
TCAV	Testing with Concept Activation Vectors

TFA	Two-stage Fine-tuning Approach
TP	True Positive
TPR	True Positive Rate
VGG	Visual Geometry Group
VINTER	Vision transformer supporting INTERpretability
ViT	Vision Transformer
YOLO	You Only Look Once

Glossary

anchor box	Predefined rectangular template used by some detectors to predict bounding boxes via refinement offsets.
attention mechanism	Neural-network operation that computes each output as an input-dependent weighted combination of all input elements, enabling models to capture long-range dependencies. Foundation of the transformer architecture.
backbone	Pretrained feature-extraction network whose output features are reused as input to task-specific heads.
bounding box	Axis-aligned rectangle localizing an object in an image, defined by two corners or by center, width, and height.
contrastive learning	Self-supervised objective that pulls representations of related inputs together and pushes unrelated ones apart.
data augmentation	Synthetic enlargement of a training set via label-preserving transformations such as cropping or flipping.
embedding	Dense numerical vector representing data in a learned feature space where geometric proximity often reflects semantic similarity.
episode	One simulated few-shot task: a small labeled support set plus a query set to predict.
few-shot learning	Learning paradigm in which a model must generalise to new categories from very few labeled examples per category.
fine-grained recognition	Recognition of categories that differ only in subtle visual details, e.g., bird species or product variants.
fine-tuning	Further training of a pretrained model on a smaller, task-specific dataset, typically with a reduced learning rate.
interactive learning	Setup in which a human operator provides feedback during preparation or inference to correct model behavior.

interpretability	Extent to which a model's decisions can be understood by humans, achieved through transparent design or post-hoc explanations.
meta-learning	<i>Learning to learn</i> : training across many small tasks so that the model is prepared to adapt quickly to a new one.
prototype	Representative feature vector for a category or object part, typically the mean embedding of labeled examples. class prototype Mean embedding of the support examples of one category. part prototype Mean embedding of one local part of objects of a category.
query set	Unlabeled examples that a model must classify or detect.
region proposal	Candidate image region that may contain an object, generated in the first stage of two-stage detectors.
self-supervised learning	Learning representations without manual labels by deriving a supervisory signal from the data itself.
semantic segmentation	Per-pixel category prediction without distinguishing individual object instances.
support set	Small set of labeled examples available to the model in a few-shot task.
transfer learning	Reuse of a model's learned representations on a new task, typically by fine-tuning a pretrained backbone.
transformer	Neural-network architecture built around self-attention layers that model dependencies across an entire input sequence in parallel.
zero-shot learning	Recognition of categories for which no labeled training examples were provided, often via text or attribute side information.

A Approach Supplements

This chapter provides additional implementation details and experiments that were not added to the main sections to improve readability. Each architecture has its own section.

A.1 LiViT

This section provides supplements for the LiViT architecture presented in [Section 4.1](#) and evaluated in [Section 6.1.1](#). The following briefly describes the implementation details for the ViT-B backbone and the Mask R-CNN.

A.1.1 ViT-B Backbone.

The pretrained ViT-B is taken from the `timm`¹ library. Several sets of weights are available:

- **augreg2 (current timm default).** Pretrain: ImageNet-21k (AugReg, v2). Finetune: ImageNet-1k. Resolved alias: `vit_base_patch16_224.augreg2_in21k_ft_in1k`.
- **augreg2 (explicit).** Pretrain: ImageNet-21k (AugReg, v2). Finetune: ImageNet-1k. Tag: `vit_base_patch16_224.augreg2_in21k_ft_in1k`.
- **augreg in1k-only.** Pretrain: ImageNet-1k only (AugReg, no in21k). No separate finetune. Tag: `vit_base_patch16_224.augreg_in1k`.
- **Original Vision Transformer (ViT) paper (Dosovitskiy et al.).** Pretrain: ImageNet-21k. Finetune: ImageNet-1k. Tag: `vit_base_patch16_224.orig_in21k_ft_in1k`.

Unless stated otherwise, the `vit_base_patch16_224.augreg_in1k` checkpoint is used. This is a ViT-B/16 with a patch size of 16×16 and an input resolution of 224×224 , pretrained on ImageNet-1k. The checkpoint must be pinned explicitly, since the default `vit_base_patch16_224` alias in `timm` currently resolves to `augreg2_in21k_ft_in1k`.

The model weights can be obtained with the following Python code snippet:

- **augreg in1k-only:**

```
timm.create_model("vit_base_patch16_224.augreg_in1k", pretrained=True)
```

¹ <https://pypi.org/project/timm/1.0.21>

A.1.2 Mask R-CNN

For training the Mask R-CNN [82] the Detectron2¹ framework was used. The training setup is described in Table A.1.

Table A.1: Mask R-CNN training configuration using Detectron2.

Parameter	Value
Backbone	ResNet-50 + FPN
Pretrained weights	COCO (mask_rcnn_R_50_FPN_3x.yaml)
Optimizer	SGD (momentum 0.9, weight decay $1e-4$)
Base learning rate	$1e-3$
LR schedule	Constant (SOLVER.STEPS = [])
Batch size (IMS_PER_BATCH)	2
ROI head batch size	512
Input size	224×224

A.2 BatterySort

This section provides supplements for the transfer learning method presented in Section 4.2 and evaluated in Section 6.1.2. The following briefly describes the training parameters for the Mask R-CNN. Details about the hardware and other technical information of the demonstrator used for evaluation can also be found in [P5].

A.2.1 Mask R-CNN

As in Section A.1, the Mask R-CNN [82] was fine-tuned with the Detectron2 framework. The training setup is described in Table A.2.

Table A.2: Mask R-CNN training hyperparameters

Parameter	Value
Base configuration	mask_rcnn_R_50_FPN_3x.yaml
Pretrained weights	COCO
Number of classes (foreground)	5
Images per batch	2
RoI proposals per image	256
Base learning rate	2.5×10^{-4}
Maximum iterations	900
Optimiser	SGD (default, momentum 0.9, weight decay 10^{-4})
Train/val short-edge resize	800 px (default)

¹ <https://github.com/facebookresearch/detectron2>

A.3 VINTER

This section provides supplements for the fine-grained few-shot recognition method presented in Sections 4.3 and 5.1 and evaluated in Section 6.1.3. The following briefly describes the implementation details for the different backbones.

A.3.1 Backbones

The two default configurations are summarized in Table A.3. Note that in this work *BLIP-2* refers to the image encoder only, which is an EVA-CLIP-G/14 [45]. The library used to integrate this encoder is LAVIS¹.

Table A.3: Backbones used for patch-feature extraction.

	DINOv2	BLIP-2 visual encoder
Library / hub	torch.hub (facebookresearch/dinov2)	LAVIS load_model_and_preprocess
Variant	dinov2_vitb14	blip2_feature_extractor / pretrain
Patch size P	14	14
Input $H \times W$	560×560	224×224
Patch grid $H_p \times W_p$	40×40	16×16
Embedding dim. D	768	1408
# layers	12	39
Heads N_{heads}	12	16
Head dim. d_k	64	88

A.3.2 Lightweight Modules

The three trainable modules of the pipeline (Foreground Estimation, Part Separation, Part Similarity) and the optional Part Reweighting are each implemented as a single `torch.nn.Linear` layer whose parameters are fit offline on the cached patch-feature dataset and then frozen. Their setup is summarized in Table A.4.

A.3.3 Concept Activation Vectors

This section extends the examination of the concept activation vectors that form the basis for the interactive interfaces of the architecture presented in Section 5.1 and evaluated in Section 6.1.3. Table A.5 provides the complementary texture concept set for the color concept set already analyzed and presented in Table 6.9.

¹ <https://github.com/salesforce/LAVIS>

Table A.4: Lightweight modules of the VINTER pipeline for fine-grained recognition.

	Foreground	Part Separation	Part Similarity
Output map	$\hat{\mathbf{M}}_{\text{fg}}$	$\hat{\mathbf{M}}_{\text{part}}$	$\hat{\mathbf{M}}_{\text{sim}}$
Deployed as	linear layer	linear layer	linear layer
Training	linear SVM	K-Means $\rightarrow$ centers	per-part linear SVM
Library	sklearn.svm.SVR / SVMTrainer	sklearn.cluster.KMeans	SVMTrainer
Input	$\mathbf{f}_{i,j} \in \mathbb{R}^{D_F}$	last-block $\mathbf{f}_{i,j} \in \mathbb{R}^{C_O}$	$\mathbf{f}_{i,j} \in \mathbb{R}^D$ or $\mathbb{R}^{D_F - C_O}$
Output dim.	1	N_p	N_p
Optimizer	SGD, mom. 0.9	k-means++ init	SGD, mom. 0.9
Batch size	64 patches	full dataset	64 patches
Epochs	512 (best-loss kept)	n_init="auto"	512 (best-loss kept)
Bias	$b_{\text{fg}} \in \mathbb{R}$	$\mathbf{0}$	$\mathbf{b}_{\text{sim}} \in \mathbb{R}^{N_p}$

Table A.5: Prediction performance using the texture concept bank across different facets and backbones.

Setting	Backbone	Facet	Textures										Textures		Total
			dotted	striped	banded	zigzagged	flecked	smeared	knitted	fibrous	Mean	Rank	Mean	Rank	
DINOres	t	q	68.90 ± 4.75	51.06 ± 5.93	62.58 ± 8.14	54.58 ± 1.51	72.50 ± 3.57	51.84 ± 6.61	63.29 ± 1.92	60.91 ± 2.38	60.71	20	63.84	20	63.84
		o	49.22 ± 2.99	51.56 ± 9.38	78.12 ± 5.26	59.38 ± 1.68	56.64 ± 1.49	46.09 ± 7.92	54.30 ± 5.61	73.83 ± 7.80	58.64	21	55.44	21	
	k	q	100.0	88.86 ± 5.54	91.83 ± 3.60	77.11 ± 8.52	91.22 ± 5.21	93.03 ± 2.04	95.71 ± 1.03	99.0 ± 1.4	92.10	16	90.46	16	90.9
		k	99.96 ± 0.06	91.78 ± 2.28	95.68 ± 3.78	79.15 ± 4.15	91.56 ± 6.05	95.44 ± 1.91	95.37 ± 3.04	97.44 ± 2.09	93.30	13	90.9	13	
		v	100.00 ± 0.01	92.69 ± 0.77	94.49 ± 3.19	75.28 ± 5.63	85.85 ± 6.86	97.49 ± 1.01	95.44 ± 0.92	98.18 ± 0.87	92.43	14	90.6	14	
DINO	t	q	100.0	94.76 ± 2.13	98.88 ± 1.91	76.82 ± 6.56	94.43 ± 2.91	96.21 ± 1.70	98.77 ± 0.56	97.79 ± 0.38	94.71	7	91.04	7	91.04
		o	99.2 ± 0.4	70.93 ± 1.47	91.69 ± 2.40	80.73 ± 8.76	86.98 ± 2.54	95.10 ± 0.98	88.20 ± 4.66	93.11 ± 1.85	88.24	19	84.58	19	
	k	q	98.95 ± 0.33	90.50 ± 1.04	93.20 ± 1.28	81.91 ± 5.50	85.24 ± 7.88	91.07 ± 2.50	96.44 ± 1.15	94.04 ± 0.69	91.42	18	89.07	18	90.48
		k	98.69 ± 1.41	92.18 ± 3.65	95.42 ± 2.03	87.20 ± 7.88	84.12 ± 1.87	87.21 ± 5.96	97.45 ± 0.79	94.74 ± 4.01	92.13	15	90.48	15	
		v	99.80 ± 0.13	89.18 ± 1.00	95.35 ± 0.43	86.42 ± 4.28	95.33 ± 1.38	91.39 ± 1.73	96.37 ± 1.02	95.48 ± 0.88	93.67	12	90.36	12	
DINOv2	t	q	99.56 ± 0.77	87.59 ± 0.67	96.84 ± 0.88	99.15 ± 0.74	93.47 ± 3.06	89.51 ± 1.48	98.50 ± 1.62	98.27 ± 1.09	95.36	4	80.51	4	80.51
		o	98.25 ± 0.30	86.66 ± 2.78	95.69 ± 1.90	98.52 ± 0.56	93.96 ± 1.33	89.05 ± 0.63	97.44 ± 1.15	98.45 ± 0.19	94.75	6	79.03	6	
	qo	qo	98.47 ± 0.56	88.41 ± 1.08	97.14 ± 0.70	98.70 ± 0.41	95.66 ± 0.78	89.00 ± 0.51	97.52 ± 0.94	98.58 ± 0.27	95.44	3	85.81	3	91.64
		ko	98.55 ± 0.98	90.59 ± 2.65	95.10 ± 2.93	96.71 ± 2.64	92.75 ± 5.48	90.99 ± 6.30	96.78 ± 1.21	97.76 ± 2.73	94.90	5	91.64	5	
		vo	99.63 ± 0.52	89.74 ± 1.82	97.62 ± 0.75	97.16 ± 1.35	96.18 ± 0.97	89.84 ± 0.61	97.29 ± 0.20	97.71 ± 0.43	95.65	1	89.65	1	
BLIP-2	to	to	99.58 ± 0.69	88.85 ± 0.97	97.07 ± 0.81	99.56 ± 0.07	93.04 ± 2.73	89.99 ± 2.28	98.92 ± 0.94	98.09 ± 0.70	95.64	2	80.43	2	80.43
		q	99.98 ± 0.02	88.87 ± 2.67	95.62 ± 1.19	88.69 ± 2.39	96.05 ± 1.10	92.05 ± 1.10	95.74 ± 1.38	98.41 ± 0.26	94.43	9	92.34	9	
	k	k	99.93 ± 0.06	88.59 ± 2.20	95.47 ± 0.72	90.41 ± 0.84	95.56 ± 1.28	91.01 ± 2.18	95.76 ± 1.22	98.17 ± 0.40	94.36	10	91.6	10	89.72
		v	99.96 ± 0.02	88.84 ± 1.77	96.57 ± 1.56	89.90 ± 2.01	95.48 ± 0.75	91.8 ± 2.1	95.66 ± 1.15	98.62 ± 0.31	94.60	8	89.72	8	
		t	99.91 ± 0.12	94.72 ± 2.15	95.71 ± 1.17	81.09 ± 5.64	94.26 ± 2.55	93.17 ± 1.09	96.75 ± 0.69	98.37 ± 0.24	94.25	11	91.85	11	
	o	o	99.46 ± 0.13	78.28 ± 5.79	97.81 ± 0.95	91.78 ± 5.75	87.08 ± 6.88	90.17 ± 1.86	94.80 ± 1.43	93.42 ± 2.57	91.60	17	75.51	17	75.51

A.4 DE-fine-ViT

This section provides supplements for DE-fine-ViT, presented in [Sections 4.4](#) and [5.2](#) and evaluated in [Section 6.1.4](#). The implementation details in [Table A.6](#) cover the default setup of the underlying DE-ViT detector and the additional DE-fine-ViT parameters.

A.4.1 DE-ViT Configuration

DE-fine-ViT extends the implementation of DE-ViT [\[127\]](#) with the Fine-grained Recognition Module (FRM). DE-ViT itself is built on top of the Detectron2 framework. For DE-fine-ViT the upstream DE-ViT config files [\[127\]](#) were adapted and extended where needed. The default parameters are listed in [Table A.6](#). Evaluation experiments frequently deviate from these defaults. The `overlapping_mode` has been adopted from [\[127\]](#). However, it was only activated during operation in robotic applications and disabled for evaluation.

Table A.6: DE-fine-ViT parameters and Detectron2 configuration.

Parameter	Default	Description
<i>Inference / FRM</i>		
<code>threshold</code>	0.45	τ_{conf} on DE-ViT scores
<code>topk_det</code>	1	$\text{top-}k_{\text{det}}$ (cfg.DE.TOPK)
<code>topk_roi</code>	3	$\text{top-}k_{\text{RoI}}$
<code>overlapping_mode</code>	True	NMS + same-class filter
<i>Detector / weights</i>		
<code>config_file</code>	open-vocabulary/lvis/vitb.yaml	DE-ViT / Detectron2 config
<code>rpn_config_file</code>	RPN/mask_rcnn_R_50_FPN_1x.yaml	offline RPN
<code>model_path</code>	vitb_0059999.pth	DE-ViT checkpoint
<code>dino_model_name</code>	dinov2_vitb14	backbone
<i>Detectron2 overrides</i>		
<code>cfg.MODEL.MASK_ON</code>	False	disables mask head (detection only)
<code>cfg.DE.OUT_INDICES</code>	[11]	last layer used for ViT features

A.4.2 Incremental Class Addition

Adding a new class does not retrain the detector. The labeled images are integrated into the support set, and the class and part prototypes are recomputed from the new images with the same frozen DINOv2 backbone. The remaining support set features and prototypes are cached on disk. The DE-ViT checkpoint is unchanged, so only the new prototype tensors need to be swapped in, which leads to updates on the order of seconds.

A.4.3 Fine-Grained Power Drill Dataset

This dataset was created as part of a supervised thesis [\[T4\]](#) and represents a typical scenario for assistive robots on a construction site. For instance, the robot’s mission could be to bring

all `drill_2` objects to the truck parked in front of the building. Such robotic scenarios were the subject of the i-SAMLR project (Section 6.2).

Two versions of annotations exist for this dataset. The recently updated version v2 provides bounding boxes that exclude corresponding drill cables more strictly. If not mentioned otherwise, v1 is used.

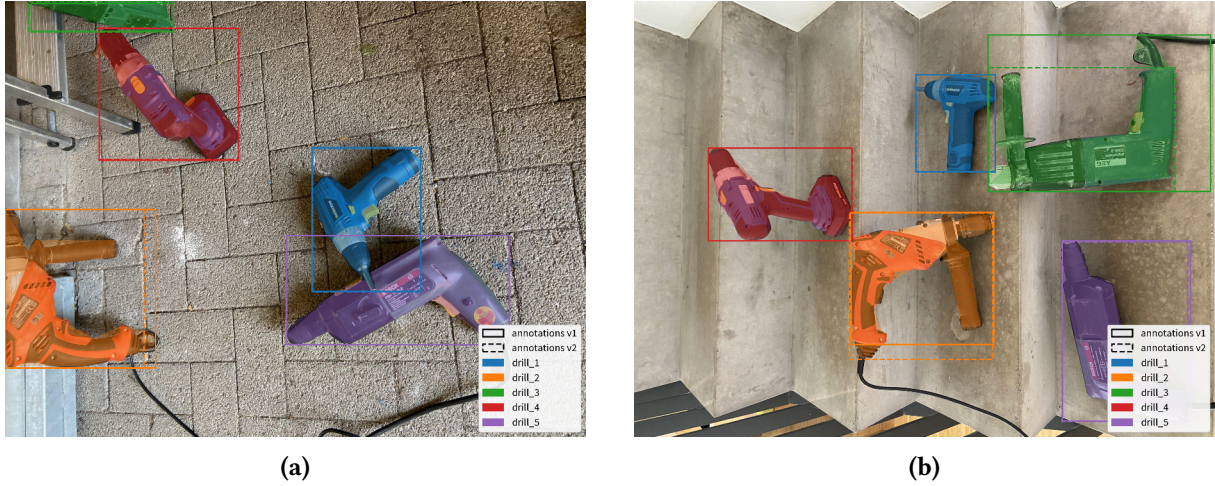


Figure A.1: Two samples from the test split of the fine-grained power drill dataset used in Section 6.1.4. In the top right region of (b), a larger portion of the drill’s cable is included in the bounding box of the corresponding v1 annotation compared to the v2 counterpart.

B Datasets for Vision Applications in Robotics

For this work, a range of datasets have been reviewed. The collection has been published as a public repository on GitHub¹.

The list of datasets covers a wide range of domains and applications, and might serve as a starting point for others when setting up their robot perception, e.g., by retrieving annotated data for relevant object categories. Rather than reporting all details in depth, it offers a brief overview so readers can quickly assess whether a dataset might be useful.

In the context of this dissertation, the datasets that could be useful for future few-shot perception research are marked as well.

¹ <https://github.com/philipsgithub/awesome-robot-vision-datasets>

