Visual representation learning and distillation across 2D and 3D tasks

Juliette Marrie

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

École doctorale : MSTII - Mathématiques, Sciences et technologies de l'information, Informatique
Spécialité : Mathématiques Appliquées
Unité de recherche : Centre de recherche Inria de l'Université Grenoble Alpes

Apprentissage de Représentations Visuelles et Distillation de Connaissances pour des Tâches en 2D et 3D

Visual representation learning and distillation across 2D and 3D tasks

Présentée par :

Juliette MARRIE

Direction de thèse :

Julien MAIRAL DIRECTEUR DE RECHERCHE, CENTRE INRIA UNIVERSITE GRENOBLE ALPES	Directeur de thèse
Diane LARLUS Naver Labs Europe	Co-encadrante de thèse

Rapporteurs :

Martial HEBERT
FULL PROFESSOR, Carnegie Mellon University
Tinne TUYTELAARS
FULL PROFESSOR, Katholieke Universiteit Leuven

Thèse soutenue publiquement le **23 mai 2025**, devant le jury composé de :

Julien MAIRAL, DIRECTEUR DE RECHERCHE, Université Grenoble Alpes	Directeur de thèse
Martial HEBERT, FULL PROFESSOR, Carnegie Mellon University	Rapporteur
Tinne TUYTELAARS, FULL PROFESSOR, Katholieke Universiteit Leuven	Rapporteure
Piotr BOJANOWSKI, SENIOR SCIENTIST, Meta	Examinateur
Gül VAROL, CHARGEE DE RECHERCHE, École des Ponts ParisTech	Examinatrice
Jean-Sébastien FRANCO, DIRECTEUR DE RECHERCHE, Centre Inria de l'Université Grenoble-Alpes	Examinateur

Invités :

Diane LARLUS
SENIOR SCIENTIST, NAVER LABS Europe
Michael ARBEL
CHARGE DE RECHERCHE, Centre Inria de l'Université Grenoble-Alpes

Abstract

Visual representation learning has advanced rapidly in recent years, driven by advances in computational resources, model design, and large-scale pretraining. However, important challenges remain in real-world applications, including learning from limited labeled data, operating under resource constraints, and learning visual representations beyond natural images. This thesis addresses these challenges through data augmentation, knowledge distillation, and 3D scene understanding.

The first part of this manuscript addresses the challenge of learning from limited labeled data, a critical issue in many real-world settings where overfitting and catastrophic forgetting remain significant concerns, even when adapting strong pretrained models. Data augmentation is widely used to mitigate overfitting and forgetting, but selecting effective transformations typically requires manual, task-specific design. We introduce a bilevel optimization approach to automatically learn optimal data augmentation policies effective for generalization in supervised learning with limited labels.

The second part of this manuscript focuses on learning under computational constraints. Large-scale models trained on billions of images exhibit strong generalization, but their substantial computational and memory demands limit real-world deployment. Techniques such as model compression, quantization, and knowledge distillation aim to reduce these demands. We establish best practices for task-specific knowledge distillation from large pretrained models such as DINOv2, showing that their strong generalization capabilities alter existing distillation strategies and highlight the need for task-aware knowledge transfer.

Finally, learning visual representations beyond natural images remains an open challenge. Most large models are trained on natural image datasets, while domains such as medical imaging, hyperspectral imaging, and 3D scene understanding lack comparable diversity. Recent breakthroughs in 3D reconstruction, including Neural Radiance Fields (NeRF) and Gaussian Splatting, enable high-quality modeling of geometry and appearance from posed images, but do not incorporate semantic understanding. We propose LUDVIG, a learning-free approach that uplifts 2D semantic features from models such as DINOv2 and CLIP into 3D Gaussian Splatting scenes.

In summary, this thesis provides insights into improving generalization with limited data, optimizing distillation from large models, and integrating semantic understanding into 3D scene representations.

Keywords

Visual representation learning, knowledge distillation, data augmentation, 3D vision.

Résumé

L'apprentissage de représentations visuelles a connu des avancées rapides ces dernières années, portées par l'augmentation des ressources de calcul, l'évolution des architectures de modèles et le pré-entraînement à grande échelle. Toutefois, des défis majeurs persistent dans les applications concrètes, notamment l'apprentissage à partir de données annotées limitées, l'adaptation à des contraintes de calcul, et l'apprentissage de représentations visuelles au-delà des images naturelles. Cette thèse aborde ces enjeux à travers des contributions en augmentation de données, distillation de connaissances, et compréhension de scènes 3D.

La première partie du manuscrit s'intéresse à l'apprentissage supervisé avec peu de données, un défi majeur en pratique où le surapprentissage peut survenir, même lors de l'adaptation de modèles préentraînés à de petits ensembles annotés. L'augmentation de données est systématiquement appliquée pour limiter ce phénomène, mais la sélection de transformations efficaces reste souvent manuelle et spécifique à la tâche. Nous proposons une approche d'optimisation bi-niveau permettant d'apprendre automatiquement des politiques d'augmentation optimales, efficaces pour la généralisation en apprentissage supervisé avec peu de données annotées.

La deuxième partie du manuscrit porte sur l'apprentissage sous contraintes computationnelles. Les modèles à grande échelle, préentraînés sur des milliards d'images, présentent de fortes capacités de généralisation, mais leurs exigences en calcul et en mémoire freinent leur déploiement dans des contextes pratiques. Des techniques telles que la compression de modèles, la quantification et la distillation de connaissances visent à réduire ces contraintes. Nous étudions comment transférer les connaissances de modèles préentraînés tels que DINOv2 pour guider l'entraînement de modèles plus légers sur des tâches spécifiques. Nous montrons que les capacités de généralisation des grands modèles remettent en question les approches classiques de distillation et soulignent l'importance de transférer des connaissances spécifiques à la tâche.

Enfin, l'apprentissage de représentations visuelles au-delà des images naturelles reste un défi ouvert. Les grands modèles sont principalement entraînés sur des jeux de données d'images naturelles, tandis que des domaines comme l'imagerie médicale, l'imagerie hyperspectrale ou la compréhension de scènes 3D manquent de données variées et en grande quantité. Des percées récentes en reconstruction 3D, telles que Neural Radiance Fields (NeRF) et Gaussian Splatting, permettent de modéliser finement la géométrie

et l'apparence de scènes 3D à partir d'un ensemble d'images d'une même scène, mais n'intègrent pas naturellement d'information sémantique. Nous proposons LUDVIG, une approche sans apprentissage qui projette des caractéristiques sémantiques 2D issues de modèles comme DINOv2 et CLIP dans des scènes 3D construites par Gaussian Splatting.

En résumé, cette thèse explore l'amélioration de la généralisation avec peu de données, via l'augmentation de données et la distillation, ainsi que l'intégration de la sémantique dans les représentations de scènes 3D.

Mots-clés.

Apprentissage de représentations visuelles, distillation de connaissances, augmentation de données, vision 3D.

Acknowledgments

I would like to thank my supervisors, Julien, Diane, and Michael, for guiding me throughout these three years. I cannot express how privileged I feel to have made this journey with you. All along, I've felt like I was in a cocoon, so happy and lucky to be among you and supported by you. Julien, thank you for trusting me and bringing me along on this journey. You've been amazing in guiding me, always pushing me to discover new areas and dive into completely new projects. I knew I could take them on with confidence, as I would be supported and guided in the best possible way. I feel so proud to have been your student, and I'll always admire you deeply.

Diane, thank you for being so involved, caring, and attentive, both to my well-being and to my research. You always found time for me, even when things were busy, and I always felt I could count on you. You were present at every step, reading, giving feedback, encouraging me, and that meant a lot. I've really grown with you, both as a researcher and as a person, and I'm really grateful for that.

Michael, my PhD would have been very different without you. I came under your wing when you were still a postdoc, and it was a very big wing, full of knowledge and kindness. I would bombard you with questions. You would stop whatever you were doing, listen, think... and come up with a genius answer. And then I'd be super happy. And even when you became a grown-up and got busier, you were always there when I came for help. For all of this, thank you.

I would also like to thank the three of you for gathering such an incredible jury. I thank all the jury members for dedicating their time to review the manuscript and attend the defense. I also thank Gul Varol for accepting to be part of my CSI (Comité de Suivi Individuel); it has been a real pleasure and honor to be followed and advised by you.

To all my colleagues at Inria and Naver Labs, thank you. It's been such a joy to share these three years with you. I would like to particularly thank Julien Z. and Boris C., my longest-serving officemates from Inria and Naver; thanks for putting up with me and for supporting me all along. I also thank Romain M., with whom I collaborated in my last project: it has been fun working and teaching with you.

I thank all my other colleagues at Inria Thoth. I thank all the permanent researchers for creating such an amazing research environment. I thank Nathalie for coordinating the team in every aspect (including the team retreats!). I thank Thomas R. for taking

care so well of the cluster and its users; you have always been there to help and are absolutely essential to the well-being of this team. I thank Alexandre Z. for the time we spent together and for your support; I thank Timothée D. for being such a great PhD twin. I thank Giacomo, Fares, Amogh, Zhou, Romain, and Jocelyn for all the frenzied foosball games we had. I thank Mathilde C. for being a mentor to me from the beginning until the end of my PhD.

I thank all my colleagues at Naver, and in particular the entire VRL team, for the stimulating research discussions and for the team events, both scientific and social, that made these years so enjoyable. I also thank the 3D Vision team for welcoming me into the Panst3R project in the final months of my PhD; collaborating with you was a true pleasure. I also thank Gustavo R. for being such a wonderful and caring colleague and friend at Naver. I thank Florent P. and Gregory R. for taking the time to discuss and guide me over my post-thesis prospects.

I also thank long-time friends. André, thank you for your constant support throughout my PhD. I thank my friends from Mines Paris for always being there.

And finally, to my family, my parents and my brother, thank you for always supporting me and trusting me in all the decisions I've made so far. I'm truly grateful.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Visual representation learning is a cornerstone of computer vision, enabling models to transform raw visual inputs into structured features suitable for tasks such as classification, detection, segmentation, and retrieval. Over the past decade, the field has undergone a major transformation: from carefully engineered feature extractors to end-to-end trainable systems powered by large-scale data and high-capacity models. This shift has led to remarkable progress, with modern visual representations now generalizing across a broad range of tasks, domains, and modalities. Yet, key challenges remain: learning effectively from limited data, adapting to computational constraints, and transferring knowledge beyond natural images. This manuscript addresses these challenges with a focus on data augmentation, knowledge distillation, and bridging 2D semantics with 3D vision for scene understanding.

1.1 Visual representation learning

Visual representation learning focuses on extracting meaningful features from raw visual data to enable semantic understanding for, *e.g.*, classification, detection, and segmentation. Over the years, this field has undergone significant evolution, driven by advances in computational power, algorithmic innovation, and shifting research goals.

Early approaches relied on hand-crafted descriptors grounded in domain knowledge. Methods such as the Scale-Invariant Feature Transform (SIFT) (Lowe, 2004), Speeded Up Robust Features (SURF) (Bay et al., 2006), and Oriented FAST and Rotated BRIEF (ORB) (Rublee et al., 2011) extracted sparse, local representations robust to scale and illumination. These were complemented by dense descriptors like the Histogram of Oriented Gradients (HOG) (Dalal and Triggs, 2005), which captured edge structures for detection tasks. For global image-level representations, approaches such as Bag-of-Words (Sivic and Zisserman, 2003; Csurka et al., 2004; Lazebnik et al., 2006) and Fisher Vectors (Perronnin and Dance, 2007; Perronnin et al., 2010) pooled local descriptors to form fixed-length representations. These features were typically paired with shallow classifiers like Support Vector Machines (SVMs) (Cortes, 1995), forming the foundation of classical vision pipelines. Local descriptors also played a central

role in geometric vision by enabling reliable matching across multiple views, a key step toward recovering 3D structure from images. Early structure-from-motion methods leveraged these correspondences to estimate camera motion and scene geometry (Tomasi and Kanade, 1992; Nistér, 2004), enabling dense reconstruction (Furukawa et al., 2010) and view synthesis (Debevec et al., 1996), and laying the foundation for modern systems like COLMAP (Schonberger and Frahm, 2016).

The emergence of deep learning marked a major paradigm shift. While early convolutional networks such as LeNet (LeCun et al., 1998) showed promising results on small datasets like MNIST, broader adoption was limited by the lack of computational resources, large labeled datasets, and effective training techniques for deep architectures (Bengio et al., 2006). As a result, deep learning only gained widespread attention when AlexNet (Krizhevsky et al., 2012) popularized CNNs, demonstrating a breakthrough in large-scale image classification enabled by GPU acceleration, ReLU activations, and the large ImageNet dataset. This success shifted the field away from hand-crafted features toward learned hierarchical representations and sparked progress in architectural design: VGG (Simonyan and Zisserman, 2015) increased depth using a simple, uniform structure, while ResNet (He et al., 2016) introduced residual connections to overcome vanishing gradients and enable very deep networks. More recently, the Vision Transformer (ViT) (Dosovitskiy, 2020) applied self-attention mechanisms (Vaswani et al., 2017) to sequences of image patches, showing that attention-based models could rival or surpass CNNs given sufficient data and computational resources.

These architectural advances were accompanied by increasingly standardized training practices, such as appropriate initialization (Glorot and Bengio, 2010; He et al., 2015) and normalization (Ioffe, 2015; Ba et al., 2016) to stabilize gradient flow, and learning rate schedules (Goyal et al., 2017) to ease early optimization. To improve generalization, approaches such as dropout (Srivastava et al., 2014) and data augmentation are critical. Data augmentation, in particular, enables the generation of diverse views of an image that preserve its semantic content while varying low-level details, thereby encouraging models to learn more robust and invariant representations. However, its effectiveness depends heavily on the choice and strength of transformations, often requiring manual tuning tailored to each task. In Chapter 3, introduced in Section 1.2, we focus on automating this selection process through bilevel optimization.

As labeled data became a bottleneck, self-supervised learning (SSL) emerged as a dominant pretraining strategy. Contrastive learning methods like SimCLR (Chen et al., 2020a) and MoCo (He et al., 2020) maximized agreement between different views of the same image, while clustering-based methods (Caron et al., 2018, 2020), self-distillation (Grill et al., 2020; Caron et al., 2021), and masked modeling (He et al.,

2022; Bao et al., 2022) introduced new ways to learn structure from unlabeled data. The result was a new generation of general-purpose models such as DINOv2 (Oquab et al., 2024), which demonstrated strong transferability across diverse tasks without task-specific supervision. Vision-language models such as CLIP (Radford et al., 2021) and BLIP (Li et al., 2022c) extended representation learning to multimodal settings, aligning visual features with textual descriptions. Specialized models like SAM (Kirillov et al., 2023) leveraged scale and tailored supervision to produce state-of-the-art results for segmentation and dense prediction. Together, these models form a rich landscape of pretrained vision systems, each optimized for different forms of generalization.

As visual models have scaled in size and complexity, efficient deployment has become a central concern. This has led to a broad literature on model compression, aimed at reducing memory, computation, and energy use with minimal accuracy loss. Key strategies include pruning, which removes redundant weights (Han et al., 2016; Frankle and Carbin, 2019; Zhu and Gupta, 2018), quantization, which lowers numerical precision for compact inference (Jacob et al., 2018), parameter-efficient fine-tuning methods like LoRA (Hu et al., 2022), and knowledge distillation, which transfers predictive behavior from large models to smaller ones (Hinton et al., 2015). In Chapter 4, introduced in Section 1.3, we outline best practices for knowledge distillation in supervised learning.

Alongside advances in 2D vision and language, 3D scene understanding has seen rapid progress. Seminal works in multi-view geometry and structure-from-motion (Schonberger and Frahm, 2016; Furukawa et al., 2010) enabled large-scale 3D reconstruction from unstructured image collections, while differentiable rendering (Loper and Black, 2014; Liu et al., 2019) introduced a paradigm shift by allowing gradients to flow through the rendering process. By enabling joint optimization of geometry and appearance from image supervision, differentiable rendering set the stage for neural volumetric representations—and paved the way for Neural Radiance Fields (NeRF) (Mildenhall et al., 2021), which model scenes as continuous fields of density and view-dependent color. More recently, Gaussian Splatting (Kerbl et al., 2023) marked another breakthrough by introducing a point-based alternative with real-time rasterization, directly optimizing anisotropic 3D Gaussians to achieve high-fidelity rendering without the computational overhead of volumetric ray marching. These breakthroughs, together with the success of large-scale pretrained 2D vision models such as DINO, CLIP and SAM, have fueled efforts to enrich 3D representations with high-level semantics. However, existing methods typically require iterative optimization to transfer 2D features into 3D, which can be computationally expensive. In Chapter 5, introduced in Section 1.4, we propose a simpler, learning-free approach that uplifts 2D visual representations into Gaussian Splatting scenes, enabling efficient, high-quality rendering and semantic understanding.

1.2 Improving generalization with limited labels

Context. Improving generalization on supervised tasks often requires careful tuning of hyperparameters, thoughtful model architecture design, and effective data augmentation strategies. Data augmentation improves generalization by increasing the diversity of the training set, helping models learn invariances and become more robust to input perturbations (Shorten and Khoshgoftaar, 2019). The success of AlexNet (Krizhevsky et al., 2012), for instance, relied heavily on data augmentation techniques such as random cropping, flipping, and PCA-based color perturbations. Since then, augmentation has remained central in visual recognition tasks. However, selecting appropriate augmentations and tuning their parameters, such as their intensity or frequency, remains challenging. Standard augmentations designed for natural images may not transfer well to other modalities or domains, where it is often unclear which transformations are beneficial. Manual tuning is both time-consuming and prone to suboptimal choices. This motivates the need for automated augmentation policy search methods that adapt to the specifics of a task or dataset without relying on hand-crafted priors.

AutoAugment (Cubuk et al., 2019), one of the earlier approaches for automatic data augmentation policy search, formulates the problem as a reinforcement learning loop that searches over discrete augmentation strategies. However, it is computationally expensive, requiring a full model retraining for each policy evaluation. More efficient alternatives were later introduced, based on population-based training (Ho et al., 2019), Bayesian optimization (Lim et al., 2019), and bilevel optimization (Hataya et al., 2020; Li et al., 2020; Zhang et al., 2020; Hataya et al., 2022).

Bilevel optimization frames the search for optimal augmentation policies as a nested problem: minimizing validation loss (outer problem) under the constraint that model weights are optimized on augmented data (inner problem). This approach is closely related to hyperparameter optimization and neural architecture search (Franceschi et al., 2018; Zoph, 2016), and is often tackled through unrolled optimization (Arbel and Mairal, 2022), where the inner optimization is approximated with a fixed number of gradient steps. However, as training progresses, this approximation may lead to instability because model weights become suboptimal for the current augmentation policy. Moreover, many augmentations are non-differentiable in their parameters, requiring the use of stochastic gradient estimators such as REINFORCE (Williams, 1992) or RELAX (Grathwohl et al., 2018), which respectively suffer from high variance or bias.

To reduce complexity, prior approaches learn augmentations on top of a fixed set of default transformations known to be beneficial, such as random resizing or cropping, and color jittering. Yet under these assumptions, simple alternatives such as a grid

search (Cubuk et al., 2020) or even a fixed policy with no search at all (Müller and Hutter, 2021) can rival prior bilevel approaches. These default transformations are typically tailored to natural images, which is also the domain where such methods are most commonly evaluated, and they may not apply to other modalities.

Contributions. To address these challenges, we propose *SLACK: Stable Learning of Augmentations with Cold-start and KL Regularization*, a bilevel optimization approach that learns optimal data augmentation policies without relying on prior domain knowledge. SLACK stabilizes policy updates via a Kullback–Leibler (KL) regularization term that defines a trust region around a reference policy. This reference policy is periodically updated through a multi-stage optimization process which we refer to as *cold-start*. Our contributions can be summarized as follows:

1. We introduce a simple and interpretable augmentation policy model that learns how often each transformation should be applied, as well as an upper bound on its magnitude (e.g., rotation angle or color intensity).
2. We propose a multi-stage algorithm with a cold-start strategy and KL regularization to improve stability in the optimization of the bilevel problem.
3. In contrast to prior works, our model learns augmentation policies *from scratch*, without hard-coding any transformation in the policy.

SLACK learns competitive augmentation policies across diverse domains, matching the performance of prior methods that rely on hard-coded transformations such as color jittering or random resizing and cropping.

Limitations and insights. While this study has shown that the bilevel problem underlying automatic augmentation policy search can be effectively tackled, it also revealed some of the limitations of current research perspectives. One key observation is that with networks of sufficient capacity, classical transformations rarely have a negative impact, even when not directly related to domain-specific invariances (*e.g.*, applying color transformations to sketches). We also found that applying sufficiently strong and diverse transformations was often more important than relying on sophisticated augmentation policies. In many cases, only marginal gains were observed over augmentation strategies commonly used in computer vision, even in domains beyond natural images, such as sketches or paintings. Naturally, careful augmentation design is still essential in certain cases. For example, for tasks with inherently non-invariant training data (*e.g.*, fixed aspect ratios or acquisition conditions), enforcing invariances may hinder the learning of helpful inductive biases (*e.g.*, object size).

This chapter is based on the following publication.

Juliette Marrie, Michael Arbel, Diane Larlus, and Julien Mairal.
SLACK: Stable Learning of Augmentations with Cold-start and KL Regularization.
In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023.

1.3 Supervised learning under resource constraints

Context. Recent large pretrained visual models demonstrate robust generalization across diverse computer vision tasks (Li et al., 2022b; He et al., 2022; Fang et al., 2023; Oquab et al., 2024). Developed by leveraging substantial computational resources, these models are trained on enormous (often internal) sets of visual data, enabling them to learn rich visual representations. Such models exhibit remarkable transfer performance on downstream tasks with frozen features, achieving competitive results through simple linear probing.

However, the size of the best-performing models often poses limitations for various real-world applications, both in terms of inference time and memory usage, especially in settings with limited memory or compute. As pretrained models continue to scale, compressing them while preserving their performance becomes a central concern. Yet despite advances in compression strategies, there remains a significant loss of information when transferring knowledge from large to small models. Even state-of-the-art compact models, such as DINOv2's ViT-S (Oquab et al., 2024), fall short of matching the representation quality of their larger counterparts. This gap is evident, for instance, in Fig. 1.1, which shows PCA projections of patch embeddings from ViT-g and ViT-S on the ADE20K semantic segmentation dataset. While ViT-g's embeddings exhibit decent class-wise clustering, ViT-S's representations appear far more entangled, with no clear structure emerging in the principal components.

This challenge highlights a broader difficulty in training efficient small models that generalize well across tasks. On the architecture side, efforts have focused on compact yet expressive designs. ConvNeXt (Liu et al., 2022b; Woo et al., 2023), for example, revives convolutional networks using transformer-inspired principles, while MobileViT (Mehta and Rastegari, 2022) integrates local inductive biases with attention-based modeling. Still, even with such innovations, small models trained from scratch often struggle to match the versatility and robustness of large pretrained transformers.

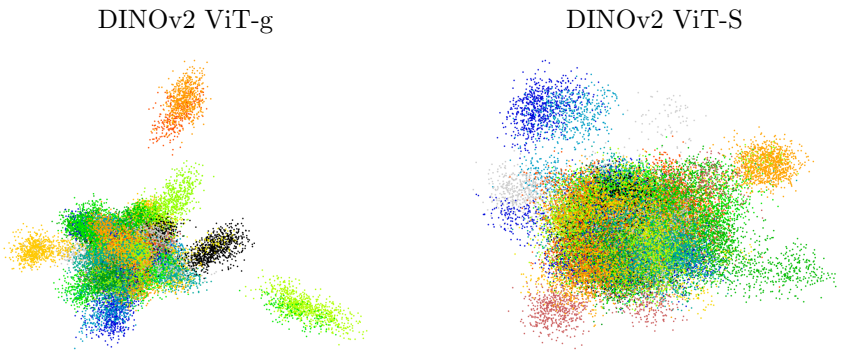


FIGURE 1.1: PCA of patch embedding representations from DINOv2's ViT-g and ViT-S pretrained models (Oquab et al., 2024) on ADE20K (Zhou et al., 2017). Classes are better clustered after distillation than after finetuning.

A promising approach for memory-efficient supervised training is to adapt large pretrained models to specific tasks without updating all of their parameters. Adapter-based methods offer a compelling alternative to full finetuning, enabling task-specific adaptation by modifying only a small subset of weights. Techniques such as adapter layers (Houlsby et al., 2019; Chen et al., 2022b), which insert small bottleneck modules between the layers of a frozen backbone, low-rank adaptation (LoRA) (Hu et al., 2022), which injects trainable low-rank matrices into existing weight projections, and IA^3 (Liu et al., 2022a), which introduces per-layer trainable scaling vectors into attention and feedforward blocks, all achieve this goal with minimal overhead. These lightweight strategies preserve most of the generality of the original model while providing sufficient task alignment, making them attractive for rapid deployment or resource-limited training. While adapter-based methods significantly reduce memory and compute requirements during training, they still rely on the full pretrained model at inference—unless combined with additional compression techniques such as quantization.

An alternative approach for directly training a small model is to transfer task-aligned knowledge from a large adapted model—typically through the distillation paradigm introduced by Hinton et al. (2015). In this formulation, a teacher trained on the task transfers representations aligned with the downstream objective, typically using a Kullback–Leibler divergence loss on the output logits, modulated by a temperature parameter. Constructing a task-specific teacher now typically requires finetuning the pretrained model on the target task, which has become standard practice in both natural language processing (Jiao et al., 2020) and vision (Huang et al., 2023).

This raises a key question: is full finetuning of the teacher necessary to perform effective task-specific distillation? Or could lighter adaptation—such as linear or shallow

probing—offer comparable or even superior guidance, while preserving more of the generality of the original model? Beyond reducing computational cost, such lightweight adaptation may help avoid overfitting and maintain broad capabilities.

Another critical axis in this context is data augmentation. It is well established that augmentation improves distillation performance, especially when labeled data is scarce (Beyer et al., 2022; Stanton et al., 2021; Wang et al., 2022). Recent generative models, such as latent diffusion models (Rombach et al., 2022), offer powerful tools to generate synthetic training samples (Trabucco et al., 2023; Azizi et al., 2023), often conditioned on class labels. However, distillation does not necessarily require ground-truth labels, suggesting that class-agnostic or label-free augmentation strategies may be particularly effective for enhancing knowledge transfer.

Contributions. In Chapter 4, we revisit current best practices for task-specific distillation, supported by extensive experiments. Our findings support a simple, memory-efficient approach for transferring task-specific knowledge from a large pretrained model to a smaller model, either pretrained or trained from scratch. Our observations are the following:

1. *Probing can yield better teachers than finetuning.* The remarkable adaptability of recent large-scale pretrained models, such as DINOv2, challenges the need for finetuning the teacher, which is standard practice in prior research on task-specific distillation (Jiao et al., 2020; Sun et al., 2019; Touvron et al., 2021; Beyer et al., 2022; Huang et al., 2023).

2. *Task-specific distillation complements task-agnostic distillation.* Task-specific distillation allows transferring task-specific knowledge, leading to better representations compared to simply finetuning the student after task-agnostic distillation, as illustrated in Fig. 4.1.

3. *Teachers do not need to be as accurate as their students.* This observation generalizes conclusions from early works (Yuan et al., 2020; Furlanello et al., 2018), conducted with teacher/student CNN models trained from scratch on a task.

4. *Small models can directly learn from much larger ones.* Prior works suggest that a large capacity gap between teacher and student hinders distillation, and employ a middle-sized 'teacher assistant' to learn from the large model and teach the small one (Jiao et al., 2020; Mirzadeh et al., 2020; Wang et al., 2020). Our results suggest that this assumption does not hold universally.

5. *Diffusion models can be effectively leveraged as data augmentation for distillation without relying on class information.* To bypass the need for class information

in Stable Diffusion, this work leverages a diffusion model that generates *mixed* images conditionally on multiple images provided as input, taking inspiration from the classical Mixup augmentation.

This chapter is based on the following publication.

Juliette Marrie, Michael Arbel, Julien Mairal, and Diane Larlus.
On Good Practices for Task-Specific Distillation of Large Pretrained Visual Models.
In **Transactions on Machine Learning Research**, 2024

1.4 Integrating semantics into 3D representations

Context. Neural scene representations have progressed rapidly in recent years, with models like NeRF (Mildenhall et al., 2021) and Gaussian Splatting (Kerbl et al., 2023) offering photorealistic view synthesis from multi-view images. NeRF introduced a powerful framework for jointly optimizing geometry and appearance via volumetric rendering, while Gaussian Splatting replaces the implicit volumetric field with an explicit set of anisotropic 3D Gaussians, enabling real-time rendering without sacrificing visual fidelity. However, integrating semantic understanding into these representations remains challenging: such models are primarily designed for reconstruction and do not encode high-level semantics.

Before the advent of differentiable rendering, enriching 3D content with semantics typically relied on projecting 2D predictions onto reconstructed surfaces, for example by transferring segmentation masks onto RGB-D meshes (Hermans et al., 2014; Dai et al., 2017) or fusing frame-wise CNN predictions into a globally consistent 3D map using SLAM (McCormac et al., 2017). These methods were constrained by the quality of 3D reconstruction and lacked a unified mechanism for leveraging large-scale visual supervision. Differentiable rendering changed this landscape by enabling end-to-end optimization of 3D structures using 2D supervision. This not only paved the way for photorealistic rendering with NeRF and Gaussian Splatting, but also opened the possibility of distilling high-level features from pretrained 2D models such as DINO (Caron et al., 2021), CLIP (Radford et al., 2021), or SAM (Kirillov et al., 2023) into 3D representations (Tschernezki et al., 2022; Kerr et al., 2023).

Building on this idea, recent work has explored uplifting 2D features into 3D scenes to enable tasks such as language-guided object retrieval (Kerr et al., 2023; Liu et al., 2023b; Zuo et al., 2024), scene editing (Kobayashi et al., 2022; Chen et al., 2024; Fan et al., 2023), and semantic segmentation (Cen et al., 2023c; Ye et al., 2024; Ying et al.,

2024). However, most of these approaches rely on iterative optimization to learn scene-specific neural fields or 3D Gaussian features by minimizing a reprojection loss. This process is computationally expensive and memory-intensive, especially when distilling high-dimensional features across millions of 3D points.

To mitigate this cost, several strategies have been proposed, including dimensionality reduction (Qin et al., 2024; Zhou et al., 2024), multi-resolution embeddings (Zuo et al., 2024), and learned feature upsamplers (Zhou et al., 2024). Others use SAM masks (Cen et al., 2023c; Kim et al., 2024; Ye et al., 2024), which require expensive image queries and complex consistency constraints across views. While effective, these methods introduce additional architectural and computational overhead and are not easily adaptable to new tasks. Furthermore, optimizing features only through reprojection raises concerns about the spatial validity of the resulting 3D representation. Without explicit consistency in 3D, the Gaussians may lack structure suitable for manipulation or reasoning. This issue has been highlighted in recent works (Wu et al., 2024b; Jun-Seong et al., 2025), which report degraded localization and generalization due to weakly grounded features.

Contributions. In Chapter 5, we demonstrate that a simple, learning-free process is highly effective for uplifting 2D features or semantic masks into 3D Gaussian Splatting scenes. This process, which can be viewed as an 'inverse rendering' operation, is both computationally efficient and adaptable to any feature type. We apply it for uplifting visual features from DINOv2 (Oquab et al., 2024; Darcet et al., 2024), semantic masks from SAM (Kirillov et al., 2023) and SAM2 (Ravi et al., 2024), and visual features from CLIP (Ilharco et al., 2021), enabling language-driven applications. Then, we show that a graph diffusion mechanism (Kondor and Lafferty, 2002; Smola and Kondor, 2003) is helpful for feature refinement in 3D scenes. This mechanism is rooted in spectral graph theory and spectral clustering (Belkin and Niyogi, 2001; Shi and Malik, 2000; Meila and Shi, 2000). In the context of our work, it transforms coarse segmentation inputs, such as scribbles or alignment scores between visual features and a text query, into accurate 3D segmentation masks without the need for segmentation models such as SAM. When evaluated on segmentation and open-vocabulary object localization tasks, our method matches state-of-the-art performance while being significantly faster than previous optimization-based approaches.

1. *A learning-free uplifting process for 2D features and semantic masks.* Our method integrates seamlessly with Gaussian Splatting and achieves competitive results on tasks such as semantic segmentation.

2. *Graph diffusion for 3D segmentation.* We propose a graph diffusion mechanism

that leverages 3D geometry and uplifted DINOv2 features to transform coarse inputs like scribbles or CLIP alignment scores into precise 3D segmentation masks, without relying on pretrained segmentation models like SAM.

3. *Combining 3D graph diffusion and 2D SAM segmentation.* We show that segmenting with SAM based on 3D scores obtained through graph diffusion achieves state-of-the-art results on open-vocabulary object segmentation tasks.

This chapter is based on the following publication.

Juliette Marrie, Romain Menegaux, Michael Arbel, Diane Larlus, and Julien Mairal. *LUDVIG: Learning-free Uplifting of 2D Visual Features to Gaussian Splatting Scenes.* **arXiv preprint arXiv:2410.14462**, 2024.

Chapter 2

Related Work

This chapter provides context for the research questions presented in Chapter 1. We begin with 2D representation learning (Sec. 2.1), reviewing early approaches based on handcrafted features and the shift toward learned methods with deep networks. Next, we discuss foundation models (Sec. 2.2), which have emerged as a central paradigm for learning transferable representations from large-scale datasets. We then turn to 3D representation learning (Sec. 2.3), covering different forms of 3D representation, multi-view reconstruction, and radiance fields. Finally, we describe the concept of knowledge distillation (Sec. 2.4), leveraged in Chapters 4 and 5 of this manuscript.

2.1 2D representation learning

2.1.1 Handcrafted features

Before the rise of deep learning, 2D image representation relied heavily on hand-designed features tailored to capture visual cues such as edges, textures, and local patterns. These representations were often combined with shallow classifiers and formed the core of early computer vision pipelines.

Local descriptors. Techniques such as Scale-Invariant Feature Transform (SIFT) (Lowe, 2004) and Histogram of Oriented Gradients (HOG) (Dalal and Triggs, 2005) extract local features that are robust to geometric transformations. SIFT computes descriptors at interest points that are invariant to scale, rotation, and illumination, making it particularly effective for image matching and retrieval. HOG (Histogram of Oriented Gradients) divides an image into small spatial regions and computes histograms of gradient orientations, providing a representation well-suited for structured object detection tasks, such as pedestrian detection.

Global and statistical representations. To obtain global image-level representations, local descriptors such as SIFT are aggregated using statistical models. Bag of Words (BoW) (Sivic and Zisserman, 2003; Csurka et al., 2004; Lazebnik et al., 2006) clusters local descriptors using unsupervised learning (typically k-means), and represents each image as a histogram over the learned visual vocabulary. Fisher Vectors

(FV) (Perronnin and Dance, 2007; Perronnin et al., 2010) go further by modeling the distribution of descriptors with a generative model—often a Gaussian Mixture Model trained via Expectation-Maximization—and encoding higher-order statistics. Both BoW and FV have been successfully applied to image classification (Csurka et al., 2004; Perronnin and Dance, 2007; Perronnin et al., 2010) and retrieval (Sivic and Zisserman, 2003; Jégou et al., 2010). These representations are often used in combination with linear classifiers such as Support Vector Machines (Cortes, 1995).

Limitations. Despite their success, hand-crafted features require manual design and domain-specific knowledge. Their limited expressiveness makes it difficult to capture high-level semantics or generalize across domains. These limitations paved the way for learned representations with deep networks, which gradually replaced hand-designed pipelines in most computer vision tasks.

2.1.2 Deep learning

Architecture. The shift from hand-crafted features to deep learning was enabled by a series of architectural advances—most notably convolutional neural networks (CNNs) (LeCun et al., 1998)—which introduced strong inductive biases such as local connectivity and translation equivariance, and enabled end-to-end learning of visual representations. CNNs first demonstrated their potential on small-scale tasks, such as handwritten digit recognition on MNIST (LeCun et al., 1998), but adoption remained limited due to constraints in data, compute, and training stability. The breakthrough came with AlexNet (Krizhevsky et al., 2012), which leveraged large-scale data (ImageNet), GPU acceleration, ReLU activations, and data augmentation to significantly outperform classical approaches. This success popularized deep learning for vision and spurred rapid architectural progress: VGG (Simonyan and Zisserman, 2015) showed that performance could be improved by increasing depth with a uniform structure, while ResNet (He et al., 2016) introduced residual connections that addressed vanishing gradient issues, enabling the training of much deeper networks.

A major architectural shift occurred with the introduction of Vision Transformers (ViTs) (Dosovitskiy, 2020), which replaced convolutional operations with self-attention mechanisms that model long-range dependencies across image patches. Adapted from transformers in natural language processing (Vaswani et al., 2017), ViTs reduced the reliance on spatial inductive biases and instead leveraged large-scale pretraining to learn general-purpose representations. Their scalability and flexibility have made them the backbone of most modern foundation models, that we review in Section 2.2.

In response, ConvNeXt (Liu et al., 2022b; Woo et al., 2023) revisited CNN design in light of transformer-era practices, incorporating elements such as large kernel sizes, GELU activations, and layer normalization. This resulted in a convolutional architecture that excels in dense prediction tasks, such as detection and segmentation, where strong spatial inductive biases offer a distinct advantage over ViTs. However, ViTs remain the dominant choice for large-scale pretraining, particularly in multimodal settings, due to their scalability, architectural simplicity, and compatibility with transformer-based learning frameworks.

Optimization. The success of learned representations stems from the ability to efficiently optimize large-scale models. Backpropagation provides the core mechanism for computing gradients in deep networks using the chain rule. Stochastic Gradient Descent (SGD) has been widely used for training deep models due to its computational efficiency on large datasets. Variants like Adam (Kingma and Ba, 2015), which adapt learning rates based on first and second moments of gradients, have improved convergence speed and stability. Learning rate schedules, such as cosine decay or warm restarts (Loshchilov and Hutter, 2017), have further enhanced performance by adapting to different stages of training. Efficient optimization remains critical for scaling deep models to high-capacity architectures and large datasets.

Data augmentation. Data augmentation has been a crucial component of deep learning since its emergence. Already in AlexNet (Krizhevsky et al., 2012), simple augmentations such as random cropping, horizontal flipping, and PCA-based color shifts were key to preventing overfitting and improving generalization on ImageNet. Since then, basic transformations like color jittering, resizing and cropping, CutMix, and MixUp (Zhang et al., 2018) have remained standard tools to increase dataset diversity and model robustness. To reduce the reliance on manual augmentation tuning, automatic augmentation methods have been developed. AutoAugment (Cubuk et al., 2019) uses a reinforcement learning controller to search over discrete transformation policies, but retraining the model at each iteration makes it computationally expensive. Subsequent approaches have improved efficiency through population-based training (Ho et al., 2019) or gradient-based bilevel optimization (Hataya et al., 2020; Liu et al., 2021). However, bilevel formulations often rely on approximations that can lead to unstable training dynamics. In Chapter 3, we propose an approach for learning data augmentation policies that stabilizes the optimization of the bilevel problem through entropy-based regularization and a cold-start strategy.

More recently, generative models have opened new possibilities for data augmentation. Latent diffusion models (Rombach et al., 2022) generate synthetic training

data from textual prompts, improving performance on classification and segmentation tasks (Trabucco et al., 2023; Azizi et al., 2023). Image-to-image diffusion models, such as Palette (Saharia et al., 2022a), generate task-specific variations like viewpoint changes and object deformations, improving model robustness and diversity. While prompt engineering remains challenging for complex tasks, methods using class names or real-image masks (Yang et al., 2023) have shown promise in reducing the need for manual intervention. In Chapter 4, we propose a data augmentation method for knowledge distillation, based on Stable Diffusion and inspired by MixUp, that avoids textual prompts. By interpolating directly between images in latent space, it applies seamlessly to both classification and segmentation tasks.

2.2 2D foundation models

Model pretraining has become a cornerstone of modern machine learning, enabling models to learn general-purpose representations from large-scale datasets that transfer effectively to a wide range of downstream tasks. This approach is especially valuable in scenarios with limited labeled data, where it significantly improves performance and sample efficiency. In this section, we review key pretraining paradigms for visual and multimodal models.

2.2.1 Vision models

Supervised pretraining. The early success of vision model pretraining was driven by supervised learning on large annotated datasets such as ImageNet (Deng et al., 2009), which enabled the development of strong visual representations and laid the foundation for progress in computer vision. This approach led to influential architectures like ViT (Dosovitskiy, 2020) and DEIT (Touvron et al., 2021), as well as task-specific models such as SAM (Kirillov et al., 2023), which achieves remarkable accuracy in segmentation. However, the reliance on large-scale manual annotations has motivated a shift toward self-supervised learning (SSL), which scales more naturally by learning from unlabeled data.

Contrastive learning. A key form of pretraining is contrastive learning, where models are trained to maximize agreement between different augmented views of the same image while pushing apart representations of different images. The objective is typically based on the InfoNCE loss (Oord et al., 2018). SimCLR (Chen et al., 2020a) scales this approach with large batches and strong augmentations, while MoCo (He et al., 2020) introduces a momentum encoder and memory queue to efficiently maintain a

large dictionary of negatives. These instance-level objectives laid the foundation for modern self-supervised representation learning.

Clustering-based learning. Clustering-based approaches generate pseudo-labels to uncover latent structure without direct supervision. DeepCluster (Caron et al., 2018) introduces this idea by alternating between k-means clustering of image features and training the model to predict the resulting assignments. SwAV (Caron et al., 2020) removes the need for offline clustering by introducing an online assignment mechanism based on Sinkhorn-Knopp optimization (Sinkhorn and Knopp, 1967), allowing cluster prototypes to be learned jointly with the encoder. SwAV also introduces a feature centering mechanism to stabilize training, which is later adopted by DINO (Caron et al., 2021). DINO builds upon SwAV and makes clustering implicit by training a student network to align with sharpened target outputs from a momentum-updated teacher.

Self-distillation. Self-distillation trains a model by having it learn from its own predictions, typically using a teacher-student framework where the teacher is a slowly updated version of the student. BYOL (Grill et al., 2020) introduced this paradigm by training a student to predict the output of a momentum-averaged teacher on a different augmented view, demonstrating that meaningful representations can emerge without negative samples. DINO (Caron et al., 2021) builds on this framework by applying a low-temperature softmax to the teacher outputs, encouraging the emergence of sharper and more clustered features. iBOT (Zhou et al., 2022a) combines self-distillation with masked image modeling, where the student predicts the teacher's representations for masked patches. Self-distillation has become a core principle in self-supervised pretraining, enabling label-free learning without contrastive negatives, while naturally preventing collapse through architectural asymmetry, stop-gradient, and slow teacher updates.

Masked modeling. Masked modeling originates from natural language processing, where models such as BERT (Devlin et al., 2019) are trained to reconstruct masked tokens from context. This idea was extended to vision by BEiT (Bao et al., 2022), which predicts discrete visual tokens obtained from a pretrained tokenizer, enabling a BERT-style masked image modeling objective. MAE (He et al., 2022) simplifies the formulation by reconstructing raw pixels instead, using an asymmetric encoder-decoder architecture that operates only on visible patches. iBOT (Zhou et al., 2022a) introduces feature-space masked prediction, where the model predicts continuous representations for masked patches, produced by a momentum-updated teacher network acting as an online tokenizer. This shift toward feature-space prediction enabled models to learn

directly from semantically meaningful targets without relying on handcrafted decoders or tokenizers, paving the way for more unified and scalable self-supervised objectives.

Unified self-supervised learning. DINOv2 (Oquab et al., 2024) unifies leading self-supervised pretraining methods and sets the current state of the art in vision. It integrates global feature alignment from DINO (Caron et al., 2021), patch-level masked prediction from iBOT (Zhou et al., 2022a), and the Sinkhorn-Knopp batch normalization of SwAV (Caron et al., 2020) to stabilize training. This combination enables the model to learn spatially precise and semantically structured features that transfer exceptionally well to a wide range of downstream tasks. In this manuscript (Chapters 4 and 5), we build upon DINOv2 to explore knowledge distillation across 2D and 3D tasks, for which we review relevant prior work in Section 2.4.

2.2.2 Vision-language models

Vision-language pretraining extends visual representation learning by incorporating language as an additional training signal. Several pretraining strategies have emerged, each leveraging different multimodal objectives.

Masked multimodal modeling. Early transformer-based models adopted masked modeling objectives inspired by BERT (Devlin et al., 2019), jointly encoding visual and textual inputs using masked language modeling and image–text matching (Lu et al., 2019; Li et al., 2019; Tan and Bansal, 2019; Chen et al., 2020b; Sariyildiz et al., 2020).

Contrastive vision-language learning. Contrastive approaches leverage large-scale datasets of paired images and text to learn aligned representations by bringing corresponding pairs closer and separating unrelated pairs. CLIP (Radford et al., 2021) exemplifies this approach, scaling up contrastive pretraining to massive image–text datasets to achieve strong zero-shot and open-vocabulary capabilities. Subsequent models (Jia et al., 2021; Li et al., 2022c) further improved upon CLIP by increasing data scale and refining alignment quality.

Multimodal conditioning and generation. More recently, approaches have incorporated language-conditioned generation and reasoning. Models like Flamingo (Alayrac et al., 2022) and LLaVA (Liu et al., 2023a) integrate visual encoders with language models, enabling advanced multimodal reasoning, text-conditioned generation, and interactive visual dialogue.

2.3 3D representation

2.3.1 Geometry representation

Early approaches to 3D scene representation focused on the explicit encoding of geometric structure. These methods can be grouped into voxel grids, surface representations, and point-based representations—each with different trade-offs in terms of fidelity, scalability, and rendering compatibility.

Voxel Representations. Voxel grids discretize 3D space into regular volumetric elements, supporting dense fusion and early convolutional modeling. TSDF fusion (Curless and Levoy, 1996) laid the foundation for volumetric reconstruction by accumulating depth into signed distance volumes. Voxel representations became central to early CNN-based shape recognition and multi-view reconstruction methods (Wu et al., 2015; Choy et al., 2016; Ji et al., 2017). However, voxel grids are limited by cubic memory scaling, prompting the development of sparse or hierarchical variants such as OctNet (Riegler et al., 2017) and O-CNN (Wang et al., 2017).

Surface Representations. Surface-based models dominate 3D graphics due to their compactness, interpretability, and compatibility with standard rendering pipelines. Classical methods reconstruct surfaces either as explicit polygonal meshes (Franco and Boyer, 2003) or as level sets of implicit functions—typically signed distance functions (SDFs) (Kazhdan et al., 2006; Kazhdan and Hoppe, 2013), which can then be converted to triangle meshes via Marching Cubes (Lorensen and Cline, 1987). More recently, deep learning methods such as DeepSDF (Park et al., 2019) and Occupancy Networks (Mescheder et al., 2019) have extended this paradigm by learning continuous implicit surface representations directly from data. Alternatively, other approaches directly predict explicit mesh structures from images (Groueix et al., 2018; Wang et al., 2018; Gabeur et al., 2019). Despite their effectiveness, surface-based methods, whether implicit or explicit, often struggle to represent complex geometries with varying topology, and tend to produce watertight surfaces that may hallucinate geometry in regions with missing or uncertain input data.

Point Representation. Point clouds offer a flexible, topology-free representation of 3D geometry by modeling scenes as discrete sets of 3D points without explicit connectivity. Early methods such as QSplat (Rusinkiewicz and Levoy, 2000) and Surfels (Pfister et al., 2000) introduced scalable rendering techniques by associating surface attributes (e.g., normals, color) with points and leveraging hierarchical data structures. In learning-based settings, PointNet (Qi et al., 2017a) enabled deep neural networks to operate directly on unordered point sets for classification and segmentation, later

extended by hierarchical and local-feature-based architectures like PointNet++ (Qi et al., 2017b). More recently, transformer-based models such as Point Transformer (Zhao et al., 2021) have further improved the capacity to model local and global relationships. Despite their flexibility, point-based representations often require high sampling density for accurate surface approximation and can struggle with uneven point distributions or discontinuities in surface coverage.

2.3.2 Multi-view reconstruction

Reconstructing 3D structure from multiple images has long been a central challenge in computer vision. A major breakthrough came with the development of Structure-from-Motion (SfM) and Multi-View Stereo (MVS), which jointly recover camera poses and 3D scene geometry from unstructured image sets. These classical pipelines laid the foundation for many large-scale and robust 3D reconstruction systems.

Structure-from-Motion (SfM). Structure-from-Motion (SfM) recovers 3D structure and camera motion by matching visual features across multiple views. Early theoretical work by Ullman (1979) established that 3D structure can be inferred purely from motion cues, motivating the development of practical algorithms. To handle outlier correspondences, Fischler and Bolles (1981) proposed RANSAC, which became crucial for robust pose estimation. Under an orthographic camera assumption, Tomasi and Kanade (1992) introduced the factorization method, while Nistér (2004) later solved the fully projective case with a minimal five-point algorithm. These methods, combined with the multi-view geometry framework of Hartley and Zisserman (2003) and the bundle adjustment formulation of Triggs et al. (1999), laid the foundation for classical SfM pipelines.

Building on these techniques, Schonberger and Frahm (2016) developed COLMAP, a unified and scalable SfM system that incrementally reconstructs scenes and refines them through frequent bundle adjustment. It also incorporates dense Multi-View Stereo (Furukawa et al., 2010) to recover high-fidelity 3D surfaces from the sparse SfM output. SfM remains widely used as a dependable initialization step in computer vision, providing accurate camera poses and sparse point clouds that can seed downstream tasks like dense reconstruction or localization.

Recently, Wang et al. (2024c) proposed DUSt3R, a learning-based approach for dense 3D reconstruction from uncalibrated image pairs. By directly regressing "pointmaps" (dense 3D coordinates) from raw images, it bypasses the need for explicit calibration or classical feature matching. Duisterhof et al. (2025) further extended this idea to multi-view scenarios with MASt3R-SfM, integrating image retrieval, dense matching, and

global alignment in a fully differentiable pipeline—without relying on RANSAC. This approach achieves scalable SfM by optimizing the global scene structure end-to-end, reflecting an emerging trend toward learned, flexible 3D reconstruction methods.

Simultaneous Localization and Mapping (SLAM). SLAM extends SfM to real-time settings, continuously estimating both camera pose and scene geometry. Early probabilistic methods (Thrun, 2002) used Kalman or particle filters to maintain scene consistency over time. Subsequent work addressed dynamic environments (Wang et al., 2007), while feature-based pipelines and bundle adjustment further improved robustness and accuracy (Mur-Artal et al., 2015). Building on DUSt3R, MUSt3R (Cabon et al., 2025) generalizes pointmap-based reconstruction to simultaneous multi-view predictions in a shared coordinate frame. This allows the network to operate efficiently in both offline SfM and online SLAM settings by producing thousands of pointmaps at high frame rates, enabling robust, scalable reconstruction even in dynamic scenes.

2.3.3 Radiance Fields

Classical multi-view and image-based rendering methods (Debevec et al., 1996; Levoy and Hanrahan, 1996) synthesize novel views by projecting images onto reconstructed geometry, typically requiring separate camera calibration and often suffering from blending artifacts. Early neural methods improved blending by learning how to combine reprojected images (Hedman et al., 2018), but still relied on precomputed geometry. A key shift came with *differentiable rendering*, which learns a scene representation directly by comparing rendered and real images (Loper and Black, 2014; Liu et al., 2019). Unlike mesh-based renderers, it enables smooth optimization and jointly refines geometry and appearance—laying the groundwork for Neural Radiance Fields (NeRF) (Mildenhall et al., 2021).

Neural Radiance Fields (NeRF). Neural Radiance Fields (Mildenhall et al., 2021) represented a significant breakthrough by modeling scenes implicitly as continuous volumetric fields encoded by multilayer perceptrons (MLPs). NeRF predicts density and view-dependent radiance at any 3D location, synthesizing novel views via volumetric rendering and ray marching. By avoiding explicit surfaces and hard edges, the continuous density field facilitates smooth optimization with accurate gradients, addressing limitations of mesh-based representations. However, NeRF's reliance on dense ray marching and per-scene optimization led to long training times, prompting extensive work on efficiency and compression. Tensor-based representations such as TensoRF (Chen et al., 2022a), Plenoxels (Fridovich-Keil et al., 2022), and Instant-NGP (Müller et al., 2022) greatly accelerated training and rendering while preserving

fidelity.

Gaussian Splatting. Gaussian Splatting (Kerbl et al., 2023) introduced a discrete, point-based alternative to radiance field rendering. Unlike implicit methods relying on ray marching, it explicitly represents radiance and geometry using anisotropic 3D Gaussians, which are efficiently rasterized with GPU acceleration. This enables real-time photorealistic rendering and significantly improves performance over volumetric representations. Gaussian Splatting builds upon classical point-based rendering methods such as QSplat (Rusinkiewicz and Levoy, 2000), differentiable surface splatting (Yifan et al., 2019), and neural point rendering (Kopanas et al., 2021). Its impact is reflected in recent extensions to dynamic content (Wu et al., 2024a), real-time SLAM (Matsuki et al., 2024), and semantic scene understanding, as discussed in Section 2.4.2.

2.4 Knowledge distillation

Knowledge distillation is a general paradigm for transferring knowledge from one model (the teacher) to another (the student), and was originally introduced in the context of supervised learning to improve the training of smaller models using the predictions of larger ones (Hinton et al., 2015). Since then, it has been widely adopted across a range of learning setups. In model compression, distillation allows compact models to inherit general-purpose representations from large pretrained networks (Fang et al., 2021; Wu et al., 2022; Oquab et al., 2024). In supervised learning, it helps guide small models on specific downstream tasks (Sun et al., 2019; Touvron et al., 2021; Beyer et al., 2022; Jiao et al., 2020; Huang et al., 2023). In self-supervised learning, the teacher is often a moving average of the student, providing stable targets without labels (Grill et al., 2020; Caron et al., 2021) (see Section 2.2). In continual learning, distillation is used to preserve previously acquired knowledge and reduce catastrophic forgetting (Li and Hoiem, 2017; Douillard et al., 2020; Fini et al., 2022; Van de Ven et al., 2022). Finally, distillation has been extended to cross-modality transfer, such as transferring 2D visual knowledge into 3D scene representations (Kerr et al., 2023; Cen et al., 2023b; Qin et al., 2024).

We next detail two lines of work that relate to our contributions. Section 2.4.1 discusses task-specific knowledge distillation, which we build upon in Chapter 4. Section 2.4.2 reviews distillation from 2D vision models into 3D scenes, studied in Chapter 5.

2.4.1 Distilling knowledge into smaller models

Knowledge distillation has evolved to transfer general representations produced by a large generic model into smaller ones (Abbasi Koohpayegani et al., 2020; Fang et al., 2021; Xu et al., 2022; Gao et al., 2022; Navaneet et al., 2021; Wu et al., 2022; Duval et al., 2023). There, distillation is used as a knowledge compression mechanism, which is motivated by the observation that directly pretraining small models on large amounts of data leads to underwhelming results compared to learning them by distillation from large pretrained models (Abbasi Koohpayegani et al., 2020; Fang et al., 2021; Xu et al., 2022; Wu et al., 2022; Oquab et al., 2024). In the context of self-supervised learning, it is then common to finetune distilled models on various downstream tasks, without further exploiting the teacher's knowledge (Sun et al., 2019; Touvron et al., 2021; Beyer et al., 2022). Surprisingly, only few studies have explored a task-specific distillation procedure that leverages both the teacher and the downstream task. An example is the two-stage distillation introduced in natural language processing by Jiao et al. (2020) and recently applied to vision tasks by Huang et al. (2023). Specifically, their approach involves a conventional generic distillation, followed by finetuning the teacher on a downstream task and applying a second task-specific distillation involving the finetuned teacher. In this manuscript, we find that finetuning the teacher can sometimes be suboptimal, and we advocate for a less computationally demanding approach.

2.4.2 Distilling knowledge into 3D scenes

With the rise of NeRF (Mildenhall et al., 2021) and Gaussian Splatting (Kerbl et al., 2023), a growing line of work has focused on enriching these 3D scene representations with semantics from pretrained 2D vision models. This semantic transfer—often viewed as a form of knowledge distillation—seeks to embed high-level 2D features into the 3D domain, supporting tasks such as segmentation, editing, and retrieval within reconstructed scenes. Several tasks have been addressed, such as semantic segmentation using SAM (Cen et al., 2023b; Ye et al., 2024; Kim et al., 2024), language-driven retrieval or editing using CLIP features regularized with DINO (Kerr et al., 2023; Zuo et al., 2024) or SAM (Qin et al., 2024; Ye et al., 2023), scene editing using diffusion models (Chen et al., 2024; Wang et al., 2024a), and 3D-aware finetuning (Yue et al., 2024). These works learn 3D semantic representations by minimizing a reprojection loss, which can be prohibitively expensive when distilling high-dimensional features. To address this, methods such as dimensionality reduction (Qin et al., 2024; Zhou et al., 2024) and multi-resolution scene representation (Zuo et al., 2024) have been proposed. In this manuscript, we propose an entirely parameter-free method, significantly accelerating the uplifting process while reducing memory requirements.

Chapter 3

SLACK: Stable Learning of Augmentations with Cold-start and KL regularization

Data augmentation is known to improve the generalization capabilities of neural networks, provided that the set of transformations is chosen with care, a selection often performed manually. Automatic data augmentation aims at automating this process. However, most recent approaches still rely on some prior information; they start from a small pool of manually-selected default transformations that are either used to pretrain the network or forced to be part of the policy learned by the automatic data augmentation algorithm. In this paper, we propose to directly learn the augmentation policy without leveraging such prior knowledge. The resulting bilevel optimization problem becomes more challenging due to the larger search space and the inherent instability of bilevel optimization algorithms. To mitigate these issues (i) we follow a successive cold-start strategy with a Kullback-Leibler regularization, and (ii) we parameterize magnitudes as continuous distributions. Our approach leads to competitive results on standard benchmarks despite a more challenging setting, and generalizes beyond natural images.[1].

This work, co-authored with Michael Arbel, Diane Larlus, and Julien Mairal, was published at the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2023.

[1]Project page: https://europe.naverlabs.com/slack

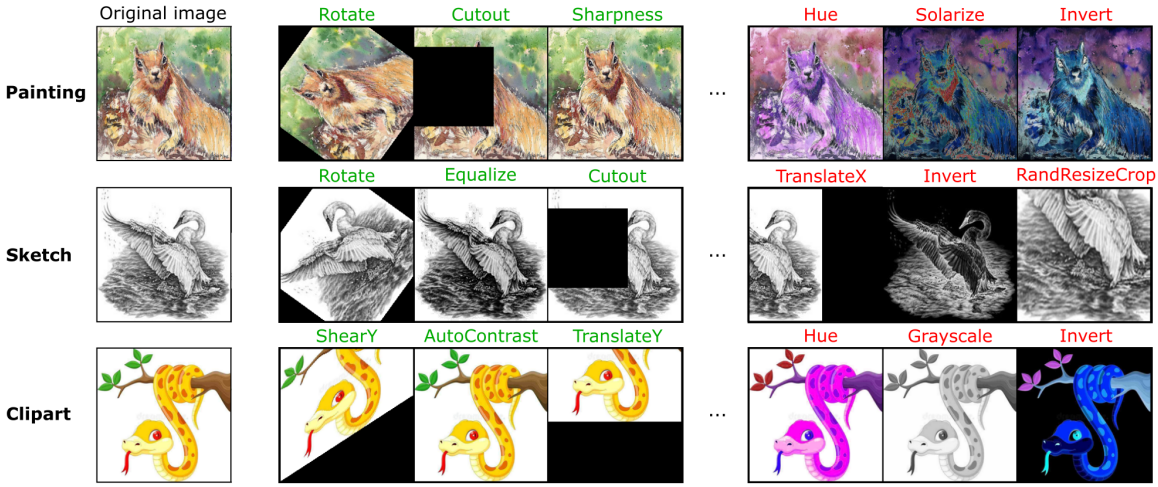


FIGURE 3.1: For different domains of the DomainNet dataset (Peng et al., 2019) (one per line), we show an image from that domain (left) and that image transformed using the three most likely (middle) and the three least likely (right) augmentations for that domain, as estimated by SLACK.

3.1 Introduction

Data augmentation, which encourages predictions to be stable with respect to particular image transformations, has become an essential component in visual recognition systems. While the data augmentation process is conceptually simple, choosing the optimal set of image transformations for a given task or dataset is challenging. For instance, designing a good set for ImageNet (Deng et al., 2009) or even CIFAR-10/100 (Krizhevsky, 2009) has been the result of a long-standing research effort. Whereas data augmentation strategies that have been chosen by hand for ImageNet have been used successfully for many recognition tasks involving natural images, they may fail to generalize to other domains such as medical imaging, remote sensing or hyperspectral imaging.

This has motivated automating the design of data augmentation strategies (Ho et al., 2019; Hataya et al., 2022; Li et al., 2020; Lim et al., 2019; Lin et al., 2019; Müller and Hutter, 2021; Wang et al., 2021b; Zheng et al., 2022). Those are often represented as a stochastic *policy* that randomly draws a combination of transformations along with their magnitudes from a large predefined set, each time an image is sampled. The goal becomes to learn strategies that effectively compose multiple transformations, which is a challenging task given the large search space of augmentations.

A natural framework for learning the parameters of this policy is that of bilevel optimization. Intuitively, one looks for the best possible policy such that a neural network trained with this policy on a training set (inner problem) generalizes well on a distinct validation set (outer problem). Optimizing the resulting formulation is challenging as

the outer problem depends on the solution of the inner problem. Classical techniques for solving this bilevel problem, such as unrolled optimization, can become highly unstable as the network weights become progressively suboptimal for the current policy during the learning process.

Moreover, augmentations are often non-differentiable in the parameters of the policy, thus requiring techniques other than direct differentiation, such as Bayesian optimization (Lim et al., 2019), gradient approximations (*e.g.* RELAX, Grathwohl et al. 2018), or the score method / REINFORCE (Williams, 1992) algorithm. While these techniques bypass the differentiability issues, they can suffer from large bias or variance. As a result, learning augmentation policies is a difficult problem whose challenges are exacerbated by the inherent instability of the optimization techniques developed to solve bilevel problems, such as unrolled optimization (Arbel and Mairal, 2022).

A standard way to improve stability and make the automatic data augmentation problem simpler is to reduce the search space. This is often achieved by learning the policy on top of "default" transformations such as Cutout (DeVries and Taylor, 2017), random cropping and resizing, or color jittering, all known to be well-suited to natural images which compose standard benchmarks such as CIFAR or ImageNet, or by discarding transformations known to be harmful such as Invert. Fixing some of the transformations and removing others mitigate the challenges inherent to learning a composition of transformations. TrivialAugment (Müller and Hutter, 2021) also shows that state-of-the-art results can be achieved on these previous benchmarks simply by directly applying the policy classically used for initializing auto-augmentation models, up to minor modifications. Moreover, all methods rely on carefully chosen ranges that constraint the transformation's magnitudes. Despite its effectiveness, manually selecting default transformations and magnitude ranges restricts the applicability of such policies to natural images and prevents generalisation to other domains.

In this paper, our goal is to choose augmentation strategies without relying on default transformations nor on hand-selected magnitude ranges known to suit common benchmarks. To achieve this objective, we first introduce a simple interpretable model for the augmentation policies which allows learning both the frequency by which a given augmentation is selected and the magnitude by which it is applied. Then, we propose a method for learning these augmentation policies by solving a bilevel optimization problem. Our method relies on the REINFORCE technique for computing the gradient of the policy and on unrolled optimization for learning the policy, both of which can result in instabilities and yield high variance estimates.

To address these issues, we introduce an efficient multi-stage algorithm with a cold-start strategy and a Kullback-Leibler (KL) regularization that are designed to improve

the stability of the process for learning the data augmentation policy. More precisely, the algorithm first pre-trains a network with a data augmentation policy uniformly sampling over all transformations. Then, each stage uses a "cold-start" strategy by restarting from the pre-trained network and performs incremental updates of the current policy.

This multi-stage approach with cold start prevents the network from becoming progressively suboptimal as the policy is updated using unrolled optimization. The KL regularization defines a trust region for the policy to compensate for the possibly high variance of gradient estimates obtained using the REINFORCE technique and encourages exploration during training, preventing collapse to trivial solutions. This regularization is inspired by proximal point algorithms in convex optimization (Rockafellar, 1976), which have also been successful in reinforcement learning tasks (Schulman et al., 2017).

By combining the regularized multi-stage approach with our interpretable model of the augmentation policies, we obtained the proposed SLACK method, which stands for *Stable Learning of Augmentations with Cold-start and Kullback-Leibler regularization*. SLACK is an efficient data augmentation learning method that is able to address the challenging bilevel optimization problem of learning a stochastic data augmentation policy without relying strongly on prior knowledge. Figure 3.1 illustrates the transformations found by SLACK to be most important / detrimental on a dataset of different domains including non-natural images.

To summarize, our contribution is threefold. (i) We propose a simple and interpretable model of the policies which allows learning both frequency and magnitudes of the augmentations. (ii) We propose a regularized multi-stage strategy to improve the stability of the bilevel optimization algorithm used for solving the data augmentation learning problem. (iii) We evaluate our method on challenging experimental settings, and show that it finds competitive augmentation strategies on natural images without resorting to prior information and generalizes to other domains.

3.2 Related work

The choice of image transformation (also known as data augmentation) has become central in the design of computer vision pipelines. To remove the burden of manual selection, automatic data augmentation strategies have been proposed (Paulin et al., 2014; Volpi and Murino, 2019). AutoAugment (Cubuk et al., 2019), one of the earliest methods, uses for instance a recurrent neural network for designing the augmentation

policy. Because such an approach requires retraining a prediction model at each iteration, it is prohibitively slow, and more efficient alternatives have been proposed. They aim at reducing the training cost using, *e.g.*, population-based training (Ho et al., 2019), Bayesian optimization (Lim et al., 2019), and more recently, gradient-based approaches based on bilevel optimization (Hataya et al., 2020, 2022; Li et al., 2020; Lim et al., 2019; Liu et al., 2021; Wang et al., 2021b; Zhang et al., 2020), relying on various gradient estimation techniques such as RELAX (Grathwohl et al., 2018) or the Score method (Williams, 1992). While the former is inherently biased, the latter is theoretically exact, but has a high variance when approximated in the context of stochastic optimization. Therefore, these approximations may lead to diverging gradient updates. Our method alleviates this by introducing a KL regularization that defines a trust-region for the policy.

Automatic augmentation using prior knowledge. Most previous works learn augmentations using a small network learned on a subset of the dataset of interest, before retraining the prediction model on a larger network using the full (augmented) data. This choice is appealing to recent gradient-based methods (Hataya et al., 2020; Li et al., 2020) as the search phase for an augmentation policy is often reduced to minutes. Nevertheless, Hataya et al. (2022); Zhang et al. (2020) have observed that policies found with such a reduced setup may be suboptimal compared to approaches exploiting full datasets for training both the augmentation policy and the prediction model. This observation was confirmed by Cubuk et al. (2020), who show that a naive grid search could actually yield state-of-the-art results when directly training on the full-size network and the full data. These results are however obtained at the expense of using strong prior knowledge: augmentation policies are applied on top of default transformations that are manually and independently chosen for each benchmark. Lately, Müller and Hutter (2021) have shown that with a few additional careful choices regarding the augmentation policies, applying a single random transformation on top of the default ones could lead to state-of-the-art results.

To avoid relying on default augmentations, DeepAA (Zheng et al., 2022) has recently proposed a greedy approach that is able to learn these transformations. Yet, learning is performed after a "pre-training" phase leveraging the usual default transformations. Moreover, while such a greedy approach simplifies the search procedure and reduces its stochasticity, the resulting computational cost is high. Instead, our approach improves stability and allows directly learning the joint probability of sampling multiple transformations, reducing the search time twofold compared to DeepAA.

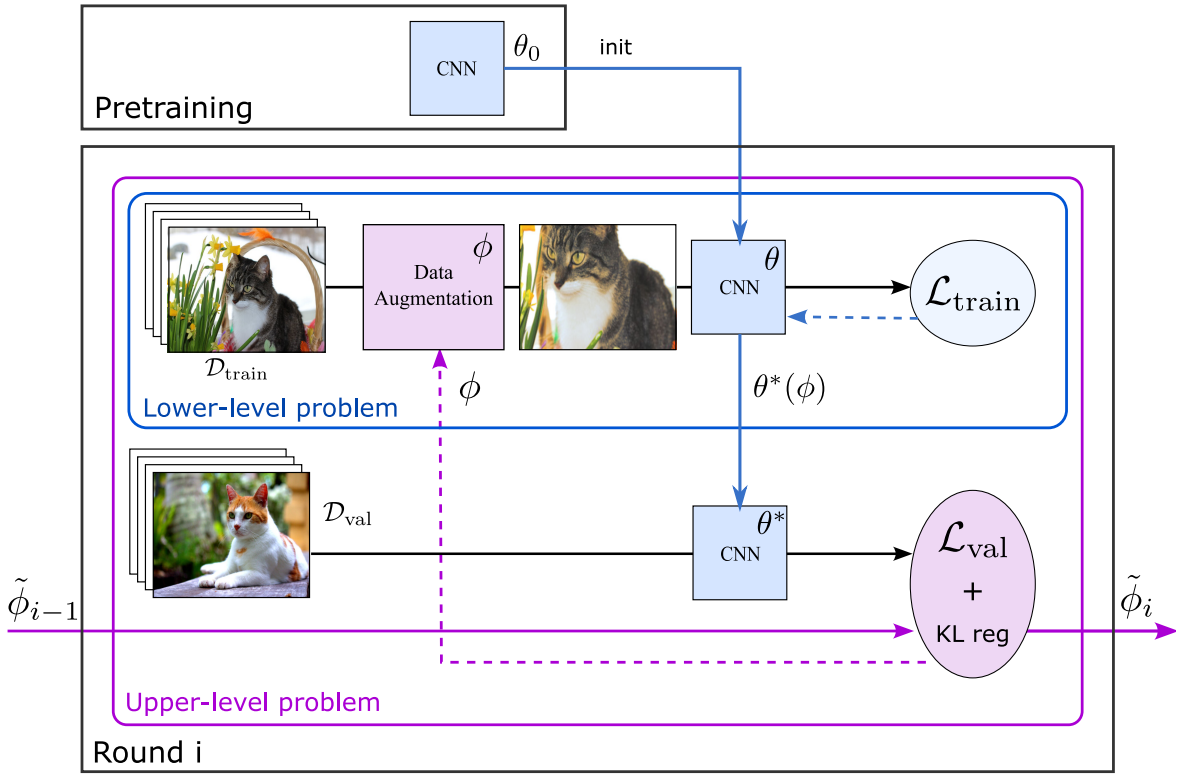


FIGURE 3.2: **Overview of our proposed SLACK method.** We learn a data augmentation policy parameterized by ϕ using bilevel optimization. The inner loop finds the optimal network parameter θ^* on images from $\mathcal{D}_{\text{train}}$. The outer loop trains on a disjoint set of images $\mathcal{D}_{\text{val}}$ using this network and finds the optimal transformation parameters ϕ. The method is enhanced with i) a cold-start strategy that structures the learning into rounds which share ϕ but restart the network from the pretrained one, and ii) a KL regularization.

3.3 Method

Our method, SLACK, defines an *augmentation policy*, which is a probabilistic model for generating data augmentations. The goal is to learn the parameters of this augmentation policy so as to improve the performance of the trained classifier on a held-out dataset. We first describe the augmentation policy (Sec. 3.3.1) and then formalize the problem of learning data augmentations with bilevel optimization (Sec. 3.3.2). We then describe our approach for solving such a bilevel optimization problem, which aims at stabilizing the optimization (Sec. 3.3.3).

3.3.1 Stochastic data augmentations policy

An augmentation function τ transforms an image x into another *augmented* image $\tau(x)$ of the same dimensions. We consider composite augmentations obtained by combining

simpler augmentations selected from a finite set $\mathcal{S} = \{s_1, \ldots, s_N\}$ of N candidate *elementary transformations*, such as rotations, translations, shearing, etc. Each elementary transformation depends on a *magnitude* parameter m that controls the strength of the transformations, for instance, the angle by which an image is rotated. Magnitudes are normalized to be in the unit interval $[0, 1]$.

Augmentation policy. We define the augmentation policy as a probabilistic model p_ϕ that generates composite augmentations given some parameter ϕ to be learned. The model generates an augmentation in three steps: (1) it samples K elementary transformations $t_1, \ldots, t_K$ from $\mathcal{S}$ according to a categorical distribution p_π of parameter π, (2) it samples values for the magnitudes $m_1, \ldots, m_K$ for each of the selected elementary transformations t_k according to a smoothed uniform distribution p_μ of parameter μ and (3) it composes the K elementary transformations to obtain the composite augmentation, with each t_k applied using its corresponding magnitude m_k. Therefore, the augmentation policy $p_\phi(\tau)$ takes the form

$$p_\phi(\tau) = \prod_{i=1}^{K} p_\pi(t_i) p_\mu(m_i|t_i), \tag{3.1}$$

where the parameters $\phi = (\pi, \mu)$ are learned jointly. Next, we describe the sampling of the transformations and of their magnitudes.

Sampling transformations. We sample elementary transformations t_k with replacement from a categorical distribution Cat_{π_k} of dimension N parameterized by a logit vector $\pi_k{:=}(\pi_{k,n})_{1\leq n\leq N}$. The probability $p_\pi(t_1, \ldots, t_K)$ of sampling the K transformations is given by:

$$p_\pi(t_1, \ldots, t_K) = \Pi_{k=1}^{K} \mathrm{Cat}_{\pi_k}(t_k), \tag{3.2}$$

where we collect all logits to form a parameter matrix π of size $K \times N$. These parameters are learned.

Sampling magnitudes. The magnitudes of each elementary transformation s_i in S are sampled from a smoothed uniform distribution between $[0, \mu_i]$ whose upper-bound μ_i is learned. More precisely, the distribution's density is defined as

$$p_{\mu_i}(m_i) = \frac{1}{\mu_i} \int_0^{\mu_i} \mathcal{N}(m_i, \sigma)(u) du,$$

where $\mathcal{N}(m_i, \sigma)$ is the Gaussian distribution of mean m_i and deviation σ. The density $p_{\mu_i}(m_i)$ approximates the uniform distribution $\frac{1}{\mu_i} \mathbf{1}_{[0,\mu_i]}$ as the deviation σ approaches

0. In practice, we set $\sigma = 0.1$ as we found it to achieve a good trade-off between smoothing and approximation.

Why a uniform distribution? We ran some ablations on previous methods (see supplementary) which suggest that a uniform sampling works on par with more elaborate sampling strategies, and that the magnitude range has more impact on the results.

3.3.2 Bilevel formulation for policy search

We consider a prediction task, such as predicting the class y of some natural image x using a model $f_\theta(x)$ with parameter θ. We are interested in finding the best policy parameter ϕ so that the prediction model f_θ, when trained using such policy on a training set $\mathcal{D}$ of input/output pairs (x, y), generalizes well on the test set $\mathcal{D}_{\text{test}}$. The problem naturally decomposes in two phases. During the *search phase*, the optimal augmentation policy p_ϕ is learned on $\mathcal{D}$. During the *evaluation phase*, the model is re-trained on $\mathcal{D}$ using p_ϕ and is then evaluated on $\mathcal{D}_{\text{test}}$. The *evaluation phase* is performed using standard optimization methods. However, the *search phase* requires solving a complex optimization problem that we describe next.

Search phase as a bilevel problem. The *search phase* naturally writes as a bilevel problem involving two interdependent losses: a lower-level loss $\mathcal{L}_{\text{train}}(\theta, \phi)$ for learning an optimal model parameter $\theta^\star(\phi)$ obtained using the augmentation policy p_ϕ and an upper-level loss $\mathcal{F}(\phi)$ for learning the policy parameter ϕ by evaluating the optimal model with parameter $\theta^\star(\phi)$. Each of these objectives is evaluated on two separate splits of the available data $\mathcal{D}$: a training split $\mathcal{D}_{\text{train}}$ for the lower-level loss and a validation split $\mathcal{D}_{\text{val}}$ for the upper-level loss. Below, we describe both losses.

Lower-level loss. We first introduce the training loss $\ell_{\text{train}}(\theta, \tau)$ when only a fixed augmentation τ is used:

$$\ell_{\text{train}}(\theta, \tau) := \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{train}}} \left[\ell(y, f_\theta(\tau(x)))\right],$$

where (x, y) is an (image,label) pair drawn from $\mathcal{D}_{\text{train}}$ and ℓ is a pointwise prediction loss (*e.g.* cross-entropy). We then define the training loss $\mathcal{L}_{\text{train}}(\theta, \phi)$ for an augmentation policy p_ϕ by taking the expectation of $\ell_{\text{train}}(\theta, \tau)$ over augmentations τ sampled according to the policy p_ϕ:

$$\mathcal{L}_{\text{train}}(\theta, \phi) := \mathbb{E}_{\tau\sim p_\phi} \left[\ell_{\text{train}}(\theta, \tau)\right].$$

Hence, for a given policy p_ϕ, the goal is to learn the optimal model parameter $\theta^\star(\phi)$ by minimizing $\mathcal{L}_{\text{train}}(\theta, \phi)$ over θ.

Upper-level loss. We first denote by $\mathcal{L}_{\text{val}}(\theta)$ the validation loss for a given model of parameter θ:

$$\mathcal{L}_{\text{val}}(\theta) := \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{val}}}\left[\ell(y, f_\theta(x))\right].$$

The validation loss $\mathcal{L}_{\text{val}}(\theta)$ is computed over the validation set $\mathcal{D}_{\text{val}}$ *without* applying any augmentation and thus provides a proxy for the performance on the test dataset. We then define the upper-level loss to be the validation loss of an optimal model $\theta^\star(\phi)$ learned using a policy p_ϕ:

$$\mathcal{F}(\phi) := \mathcal{L}_{\text{val}}(\theta^\star(\phi)). \tag{3.3}$$

While optimizing the lower-level loss is relatively standard, minimizing the upper-level loss $\mathcal{F}$ is more challenging due to the complex dependence of the optimal model parameter $\theta^\star(\phi)$ on the policy. Next, we describe our proposed algorithm for solving the bilevel problem.

3.3.3 SLACK algorithm

We propose Algorithm 1 for learning the optimal policy during the search phase. SLACK first pre-trains the prediction model using the objective $\mathcal{L}_{\text{train}}(\theta, \phi_{\text{uniform}})$ for an initial policy parametrized by ϕ_{uniform} which samples uniformly among all elementary transformations. It then performs n_{rounds} rounds to update the parameters θ and ϕ jointly using a bilevel optimization algorithm. This approach is reminiscent of the one of AutoAugment (AA) (Cubuk et al., 2019) that fully re-trains the network for each policy update. Yet, it is several orders of magnitude faster than AA, as it benefits from pre-training and from our bilevel optimization.

SLACK relies on two strategies to ensure the stability of the parameter updates during each round: a *cold-start strategy* for the prediction model and an *anchoring strategy* for the policy. The *cold-start* initializes the prediction model at the beginning of each round using a pre-trained model θ_0. *Anchoring* is achieved by encouraging the current policy to remain close to some *anchor policy* $p_{\tilde{\phi}}$. We set $\tilde{\phi}$ to the current policy parameter ϕ at the beginning of each round. During the first n_{retrain} steps of each round, the algorithm only updates the model parameter using a stochastic estimate $\hat{g}_\theta$ of $\nabla_\theta \mathcal{L}_{\text{train}}(\theta, \phi)$ while maintaining the policy fixed. Then for the last $n_{\text{total}} - n_{\text{retrain}}$ steps, the algorithm alternates between model updates and policy updates. The policy updates aim to minimize the sum of the upper-level objective $\mathcal{F}$ and an anchoring $d(p_\phi, p_{\tilde{\phi}}) := \text{KL}(p_\pi, p_{\tilde{\pi}})$ encouraging the policy p_ϕ to remain close to the *anchor policy* $p_{\tilde{\phi}}$. These updates are obtained using a stochastic estimate $\hat{G}_\phi$ along with the exact gradient of the KL regularization which admits a closed-form expression. Next we

Algorithm 1 SLACK

Initialize policy parameter $\phi \leftarrow \phi_{\textbf{uniform}}$.
Pre-training: $\theta_0 \leftarrow$ `optimize` $(\mathcal{L}_{\text{train}}(\theta, \phi))$.
for $i \in \{1, ..., n_{\text{rounds}}\}$ **do**
  Cold-start: $\theta \leftarrow \theta_0$.
  Update anchor policy: $\tilde{\phi} \leftarrow \phi$.
  for $j \in \{1, ..., n_{\text{total}}\}$ **do**
    Compute stochastic gradient $\hat{g}_\theta \approx \nabla_\theta \mathcal{L}_{\text{train}}(\theta, \phi)$
    Update θ: $\theta \leftarrow \theta - \eta \hat{g}_\theta$.
    if $j > n_{\text{retrain}}$ **then**
      Compute stochastic gradient $\hat{G}_\phi \approx \nabla_\phi \mathcal{F}(\phi)$
      Update ϕ: $\phi \leftarrow \phi - \alpha(\hat{G}_\phi + \lambda \nabla_\phi d(p_\phi, p_{\tilde{\phi}}))$.
    end if
  end for
end for

explain how we estimate the gradients $\hat{g}_\theta$ and $\hat{G}_\phi$ and discuss the effect of cold-start and KL-regularization.

Gradient estimation. Algorithm 1 requires estimating the gradient of $\mathcal{F}(\phi)$, which is challenging given the complex dependence of the upper-level loss on the policy p_ϕ through the optimal model parameter $\theta^\star(\phi)$ learned using such a policy. In line with previous works (Hataya et al., 2020; Li et al., 2020; Liu et al., 2021; Wang et al., 2021b), we approximate the optimal model parameter $\theta^*(\phi)$ with a simpler function $\hat{\theta}(\phi)$ that is easier to compute:

$$\hat{\theta}(\phi) := \theta - \eta \nabla_\theta \mathcal{L}_{\text{train}}(\theta, \phi). \tag{3.4}$$

Eq. (3.4) corresponds to one gradient step to optimize the lower-level loss starting from the current parameter θ and ϕ and using step-size $\eta > 0$. By keeping track of the dependence in ϕ and exploiting the fact that the augmentation policy p_ϕ has a score $\nabla_\phi \log p_\phi(\tau)$ that can be computed explicitly using Eq. (3.1), we can use the REINFORCE/Score method (Fu, 2006) to derive a closed-form expression for $\nabla_\phi \hat{\theta}(\phi)$ which will serve for approximating the gradient of $\mathcal{F}$:

$$\nabla_\phi \hat{\theta}(\phi) = -\eta \mathbb{E}_{\tau \sim p_\phi} \left[\nabla_\theta \ell_{\text{train}}(\theta, \tau) \nabla_\phi \log p_\phi(\tau)^\top \right].$$

Then, we approximate the upper-level loss $\mathcal{F}(\phi)$ with $\hat{\mathcal{F}}(\phi) := \mathcal{L}_{\text{val}}(\hat{\theta}(\phi))$, and its gradient $\nabla_\phi \mathcal{F}(\phi)$ with $\nabla_\phi \hat{\mathcal{F}}(\phi)$, which is obtained using the chain rule:

$$\nabla_\phi \mathcal{F}(\phi) \approx \nabla_\phi \hat{\mathcal{F}}(\phi) = \nabla_\theta \mathcal{L}_{\text{val}}(\hat{\theta}(\phi))^\top \nabla_\phi \hat{\theta}(\phi). \tag{3.5}$$

The above expression requires only first-order derivatives and matrix-vector products, which is amenable to efficient implementation using automatic differentiation softwares.

Stochastic gradient estimates. In practice, we replace all expectations by estimates on a batch of data and sampled augmentations. More precisely, to compute the approximation $\hat{g}_\theta$ to $\nabla_\theta \mathcal{L}_{\text{train}}(\theta, \phi)$, we sample B_{aug} augmentations from p_ϕ and then apply each of them to a batch of training data B_{train} from $\mathcal{D}_{\text{train}}$. Using the same batch of data and augmentation, we approximate $\hat{\theta}(\phi)$ and $\nabla_\phi \hat{\theta}(\phi)$ appearing in Eq. (3.5). Finally, we use a batch B_{val} of data from $\mathcal{D}_{\text{val}}$ to estimate $\nabla_\theta \mathcal{L}_{\text{val}}(\hat{\theta}(\phi))$ and compute $\hat{G}_\phi$, which is a stochastic estimate of $\nabla_\phi \hat{\mathcal{F}}(\phi)$ in Eq. (3.5).

Cold-start. The cold-start strategy allows to re-train the model at each round with the current augmentation policy starting from the pre-trained model. This approach is closer to the original bilevel formulation which implies finding an optimal prediction model for each policy. Initializing with a pre-trained model yields computational gain as fewer iterations are needed to optimize the model. We could instead use a *warm-start* strategy which initializes the model at each round with the learned model at the previous round. Yet we experimentally observe that such approach progressively leads to overfitting and degrades the quality of the learned policies (see supplementary).

Anchoring using KL regularization. We experimentally found that adding an anchoring $d(p_\phi, p_{\tilde{\phi}}) := \text{KL}(p_\pi, p_{\tilde{\pi}})$ with strength parameter λ when updating the policy prevents the algorithm from collapsing towards trivial policies. The anchoring affects only the categorical distribution p_π. For the magnitudes p_μ, we did not use anchoring as it is ill-defined for a uniform distribution. Instead, we simply used smaller step-sizes. Our approach takes inspiration from Proximal Policy Optimization (Schulman et al., 2017) used in the context of reinforcement learning which is known to improve policy search.

3.4 Experiments

In this section, we first briefly describe our experimental setup (Sec 3.4.1). Then we evaluate our approach on several standard benchmarks composed of natural images (Sec 3.4.2) as well as on a benchmark with other domains (Sec 3.4.3). We finally report some ablation studies (Sec 3.4.4).

3.4.1 Experimental setup

Benchmarks. We first evaluate our model on three standard benchmarks, CIFAR10 (Krizhevsky, 2009), CIFAR100 (Krizhevsky, 2009) and ImageNet-100 (Tian et al., 2019), all composed of natural images. To study how well our method generalizes beyond natural images, we also evaluate on the DomainNet dataset (Peng et al., 2019),

which contains 345 classes for 6 different domains. To ensure our protocol uses a similar number of training images for each domain, we use a reduced set of 50,000 training images for the two largest domains (real, quickdraw) and leave the remaining images for testing. For the other domains, we isolate 20% of the data for testing.

Architectures. CIFAR10/100 are evaluated with two architectures that are standard for automatic data augmentation: WideResNet-40x2 and WideResNet-28x10 (Zagoruyko and Komodakis, 2016). Unlike previous works whose search phase is only conduced with the smaller WideResNet-40x2, we search and evaluate with the same architecture, as we found it to be better (see Sec. 3.4.4). ImageNet-100 and DomainNet are evaluated with a ResNet-18 (He et al., 2016) architecture.

Transformation space. Our data augmentation search space is composed of the standard pool of 15 transformations: *Identity, ShearX, ShearY, TranslateX, TranslateY, Rotate, AutoContrast, Equalize, Invert, Solarize, Posterize, Contrast, Brightness, Sharpness, Color*. We add to this pool the transformations that previous methods usually apply by default: Cutout and RandomCrop for CIFAR, RandomResizeCrop for ImageNet, Grayscale for DomainNet. Following standard practice, when RandomResizeCrop is sampled, it is always applied first. We learn the range of its scale parameter. We do not add ColorJitter that is also applied by default in prior work for ImageNet, as it is already a mix of Brightness, Contrast and Color. However we add Hue, which is one component of ColorJitter and never applied by default. Following prior work, the magnitudes are mapped to [0,1]. After mapping, μ is initialized at 0.75 to favour exploration (see details in supplementary). We also uniformly sample magnitudes for Cutout and RandomCrop, whereas their value is hand-picked in prior work. Since the datasets are horizontally symmetric, we follow common practice and apply flip by default.

Policy search. We apply a train/val split of 0.5/0.5, meaning that half of the data is used to train the model parameters while the other half is used to learn the augmentation policy. Pre-training is done in the same setting as the evaluation (see next paragraph), except that we train only with the train data in the train/val split of the search phase. We use SGD with momentum for the optimization of the validation and training losses. For the latter, we use the same weight decay as for the final policy evaluation. We sample 8 different augmentations for computing the expectation that is needed for the stochastic gradient estimate, as detailed in Sec. 3.3.3.

Policy evaluation. We evaluate our models following the framework of TrivialAugment (Müller and Hutter, 2021). The corresponding hyperparameters can be found in the supplementary. We evaluate each policy with 4 independent runs, meaning that

our results are averaged over a total of $4 \times 4 = 16$ evaluations. Our Uniform policy (corresponding to SLACK's initialization) and our reported results on TrivialAugment are evaluated with 8 independent runs. We also report a confidence interval which contains the true mean with probability $p = 95\%$, under the assumption of normally distributed accuracies.

3.4.2 Comparison with the state of the art

We compare our method with a Uniform augmentation policy as well as many previous approaches for data augmentation, including AutoAugment (AA) (Cubuk et al., 2019), Fast AutoAugment (FastAA) (Lim et al., 2019), Differentiable Automatic Data Augmentation (DADA) (Li et al., 2020), RandAugment (RA) (Cubuk et al., 2020), Teach Augment (Suzuki, 2022), UniformAugment (Paulin et al., 2014), TrivialAugment (TA) (Müller and Hutter, 2021), and Deep AutoAugment (DeepAA) (Zheng et al., 2022).

For each method, we indicate the total number of composed transformations, and the number of hard-coded transformations among those (Tables 3.1 and 3.2). For SLACK, we evaluate the policies obtained from 4 independent search runs (each with 4 different train/val splits) to assess the robustness of our approach. We follow the same process when reproducing DeepAA on CIFAR10/100. Note that all previous methods use a single run for search, before evaluating the policy with one or multiple runs. We report 95% confidence intervals for those evaluating with multiple runs.

The supplementary provides qualitative results showing the evolution of the probability distributions over the transformations and the final estimated policies for all datasets.

CIFAR. In Table 3.1, we observe that, despite not hard-coding Cutout and RandomCrop in our policy, our method is competitive on both CIFAR10 and CIFAR100.

We found that, in general, Cutout and Rotate are selected with a high probability, while the Invert transformation is systematically discarded (see supplementary). This is consistent with the choices made in practice by prior work of adding/removing these transformations manually.

We observe a mismatch between DeepAA's reported results (Zheng et al., 2022), and those we obtain when evaluating their approach on multiple search runs, using the author's code and following their recommendations. This is likely due to the stochasticity of the search procedure.

ImageNet-100. Results for ImageNet-100 are reported in Table 3.2. We compare SLACK to our Uniform policy and to TrivialAugment (RA) and (Wide) variants, the

	# Augmentations		CIFAR10		CIFAR100	
	Total	Hard-coded	WRN-40-2	WRN-28-10	WRN-40-2	WRN-28-10
AutoAugment	4	2	96.3	97.4	79.3	82.9
Fast AutoAugment	4	2	96.4	97.3	79.4	82.7
DADA	4	2	96.4	97.3	79.1	82.5
RandAugment	4	2	-	97.3	-	83.3
TeachAugment	4	2	-	97.5	-	83.2
UniformAugment	4	2	96.25	97.33	79.01	82.82
TrivialAugment (Wide)	3	2	96.3 ± .1	97.5 ± .1	79.9 ± .2	84.3 ± .2
Uniform policy	3	0	96.1 ± .1	97.3 ± .1	78.8 ± .3	82.8 ± .3
Deep AutoAugment	6**	0*	-	97.6 ± .2	-	84.0 ± .2
Deep AutoAugment (reproduced)†	6**	0*	96.3 ± .2	97.3 ± .2	79.3 ± .4	83.4 ± .4
SLACK(Ours)	3	0	96.3 ± .1	97.5 ± .1	79.9 ± .2	84.1 ± .2

TABLE 3.1: Test accuracies on CIFAR10 and CIFAR100. For SLACKand DeepAutoAugment (reproduced) we conduct 4 independent searches, and evaluate each policy with 4 evaluation runs, meaning that we report averages over 16 evaluations. TrivialAugment and Deep AutoAugment are also evaluated with multiple evaluation runs. Results for the remaining methods are reported from the corresponding papers and based on a single run.
*: Deep AutoAutoaugment uses hard-coded transformations for pre-training.
**: Deep AutoAugment learns random flipping unlike other baselines.
† We evaluate the policies found from 4 independent search runs as we do for SLACK, using the code from the authors and following their recommendations.

	# Augmentations		ImageNet-100
	Total	Hard-coded	ResNet18
TA (RA) (Müller and Hutter, 2021)†	5	4	85.9 ± .3
TA (Wide) (Müller and Hutter, 2021)†	5	4	86.4 ± .2
Uniform policy	3	0	85.8 ± .4
SLACK	3	0	86.1 ± .2

TABLE 3.2: Test accuracies on ImageNet-100.

latter using larger magnitude ranges for its random transformation. SLACK's results lie in between both variants and improve over our Uniform policy.

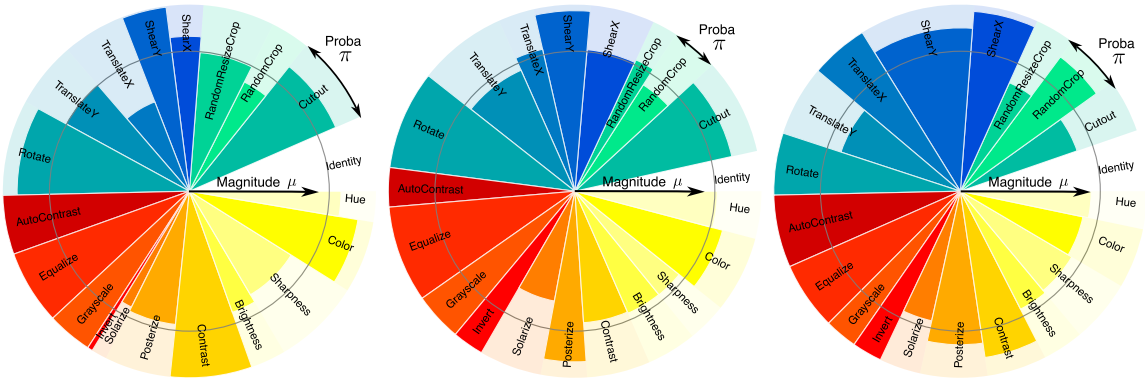
Interestingly, for ImageNet-100, we found that RandomResizeCrop is not favoured during the search phase (see supplementary), suggesting that it is not critical for ImageNet-100. Instead the performance gap between TA (Wide) and TA (RA) suggest that harder transformations are key to a better performance for this dataset.

3.4.3 Beyond natural images

For the DomainNet dataset, we compare SLACK to a Uniform policy, to the augmentations used by DomainBed (Gulrajani and Lopez-Paz, 2021) for domain generalization, and to the TrivialAugment (RA) and (Wide) methods with their ImageNet and CIFAR default settings. Results can be found in Table 3.3.

	# Augmentations		Real-50k	Quick-50k	Inforgraph	Sketch	Painting	Clipart	Average
	Total	Hard-coded							
DomainBed†	5	5	62.5 ± .2	66.5 ± 1.	26.8 ± .4	59.5 ± .4	58.3 ± .3	66.2 ± .1	57.23 ± .2
TA (RA) ImageNet†	5	4	70.9 ± .2	67.9 ± .1	**35.2 ± .2**	65.6 ± .2	64.8 ± .2	70.3 ± .2	62.43 ± .1
TA (Wide) ImageNet†	5	4	**71.6 ± .1**	68.6 ± .1	**35.4 ± .4**	**66.2 ± .2**	**65.2 ± .2**	71.2 ± .2	**63.0 ± .1**
TA (RA) CIFAR†	3	2	70.3 ± .1	68.4 ± .1	33.9 ± .3	64.1 ± .2	64.7 ± .2	70.3 ± .3	61.9 ± .1
TA (Wide) CIFAR†	3	2	71.1 ± .1	**69.3 ± .1**	34.2 ± .3	65.5 ± .3	64.8 ± .2	71.0 ± .3	62.7 ± .1
Uniform policy	3	0	70.4 ± .1	68.3 ± .1	34.1 ± .3	65.2 ± .2	64.0 ± .3	72.3 ± .2	62.4 ± .1
SLACK(ours)	3	0	71.0 ± .2	68.1 ± .2	34.8 ± .2	65.4 ± .2	64.8 ± .2	**72.7 ± .2**	62.8 ± .1

TABLE 3.3: Test accuracies on DomainNet.



(A) Painting (B) Sketch (C) Clipart

FIGURE 3.3: Policies found on DomainNet for the best search split. Gray circle: initial magnitude upper-bounds. Radius of each pie: learned upper-bounds. Size of each pie: probability of each transformation, averaged over the three composite distributions. Transformations which are parameter-free, *AutoContrast, Equalize, Grayscale, and Invert,* are displayed with maximal magnitude upper-bound.

DomainBed uses the same default transformations as TA ImageNet together with Grayscale, but with smaller magnitudes and unlike TA, does not add a random transformation. Yet it strongly overfits and performs much lower than TA. This suggests that augmentations well suited for domain generalization do not perform well on the individual tasks. TA (Wide) ImageNet consistently outperforms all other TA flavors. This further justifies the need to learn the magnitude range and to eliminate any manual range selection process.

SLACK is a close second, yet it learns the policy end-to-end. The learned policies are illustrated as pie charts in Fig. 3.3. The slices represent the probability π over the different transformations while their radius represent the corresponding magnitudes. They differ from a domain to another, suggesting that the gain compared to the initialization (*i.e.* Uniform policy) results from SLACK's ability to learn and adapt to each domain.

Search architecture	CIFAR10	CIFAR100
WRN-40-2	97.4 ± .1	83.9 ± .2
WRN-28-10 (ours)	97.5 ± .1	84.1 ± .2

TABLE 3.4: CIFAR10/100 accuracy evaluated with WRN-28-10: impact of using a smaller architecture for the search phase.

	CIFAR10		CIFAR100	
SLACKvariant	WRN-40-2	WRN-28-10	WRN-40-2	WRN-28-10
without KL	96.3 ± .1	97.1 ± .2	79.6 ± .2	83.8 ± .2
with KL (ours)	96.3 ± .1	97.5 ± .1	79.9 ± .2	84.1 ± .2

TABLE 3.5: CIFAR10/100 accuracy with/without KL regularization.

	CIFAR10		CIFAR100	
SLACKvariant	WRN-40-2	WRN-28-10	WRN-40-2	WRN-28-10
Uniform policy	96.22 ± .10	97.38 ± .05	79.07 ± .24	83.26 ± .17
μ only	96.20 ± .08	97.42 ± .05	79.22 ± .17	83.57 ± .18
π only	96.22 ± .09	97.35 ± .04	79.36 ± .11	83.45 ± .15
π and μ (ours)	96.29 ± .08	97.46 ± .06	79.87 ± .11	84.08 ± .16

TABLE 3.6: CIFAR10/100 accuracy when only learning part of the policy parameters

3.4.4 Ablation study

In this section, we evaluate our contributions and main design choices: the network architecture used for search, the KL regularization, and the benefits of learning π and μ. More ablations can be found in the supplementary. Note that hyperparameters are adjusted to each baseline included in the comparison, to make them as competitive as possible.

Network architecture for search. In prior works, the search phase (when there is one) is conducted with the smaller WideResNet-40x2 architecture for CIFAR10 and CIFAR100, and the learned policy is evaluated for both WideResNet-40x2 and WideResNet-28x10. Table 3.4 shows that for SLACK, using the same WideResNet-28x10 architecture for search and evaluation gives the best results for that architecture.

KL regularization. We compare SLACK with a flavor that does not apply KL-regularization. For the latter, we reduce the outer learning rate so that the augmentation policies with and without regularization evolve at similar speeds. Results in Table 3.5 show that our regularization is beneficial.

Joint learning of π and μ. Lastly, we study how beneficial jointly learning our augmentation parameters is compared to the initial Uniform policy and to a setting where only π or μ is learned. Results can be found in Table 3.6.

3.5 Conclusion

In this paper, we address the task of automatic data augmentation. Considering the more challenging bilevel optimization problem that arises when the search space is not reduced with default transformations, our proposed SLACK method tackles the resulting stability issues thanks to a multi-stage approach based on cold-start, coupled with a KL-regularization. Combined, they allow to reduce the variance of the gradient estimate and to better control the optimization process. We have experimentally observed that our method performs on par with recent approaches leveraging prior knowledge. It has also proved versatile enough to select domain-specific transformations when confronted to non-natural images.

Chapter 4

On Good Practices for Task-Specific Distillation of Large Pretrained Visual Models

Large pretrained visual models exhibit remarkable generalization across diverse recognition tasks. Yet, real-world applications often demand compact models tailored to specific problems. Variants of knowledge distillation have been devised for such a purpose, enabling task-specific compact models (the students) to learn from a generic large pretrained one (the teacher). In this paper, we show that the excellent robustness and versatility of recent pretrained models challenge common practices established in the literature, calling for a new set of optimal guidelines for task-specific distillation. To address the lack of samples in downstream tasks, we also show that a variant of Mixup based on stable diffusion complements standard data augmentation. This strategy eliminates the need for engineered text prompts and improves distillation of generic models into streamlined specialized networks.[1].

This work, co-authored with Michael Arbel, Julien Mairal, Diane Larlus, was published in Transactions on Machine Learning Research (TMLR) 2024.

[1]Project page: https://europe.naverlabs.com/tskd

4.1 Introduction

Recent large pretrained visual models demonstrate robust generalization across diverse computer vision tasks. Developed by leveraging substantial computational resources, these models are trained on enormous (often internal) sets of visual data, enabling them to learn rich visual representations. Such models exhibit remarkable transfer performance on downstream tasks with frozen features, achieving competitive results through simple linear probing (see, *e.g.* Li et al., 2022b; He et al., 2022; Fang et al., 2023; Oquab et al., 2024). However, the size of the best performing models often poses limitations for various real-world applications, both in terms of inference time and memory usage, especially in scenarios with constrained resources.

An essential question thus emerges: *How can we most effectively transfer the rich visual representations from these large models to a smaller architecture?* While smaller models distilled from the larger ones on a sizeable generic dataset are sometimes available (Oquab et al., 2024), is simply finetuning them to specific tasks optimal? As visual pretrained models are becoming larger, the cost of finetuning them is often out of reach for many users. It is therefore natural to ask whether a teacher trained with simple probing (linear or with a small multilayer perceptron) is sufficiently competent to guide the training of a smaller model, specialized for a given computer vision task. Finally, as distillation often benefits from data augmentation (Beyer et al., 2022), and given the effectiveness of data augmentation methods based on Stable Diffusion (Rombach et al., 2022; Saharia et al., 2022b) in supervised learning (Trabucco et al., 2023; Azizi et al., 2023; Zhou et al., 2022b), leveraging generative models for distillation seems promising. However, unlike in supervised learning where the generative model is usually conditioned by text prompts, *e.g.* , with class labels, the dependence on class information becomes questionable when used for distillation, as labels are not technically required. This raises the question of how to best leverage these models in the context of knowledge distillation.

In this paper, we study these fundamental questions. (i) We delineate optimal practices for leveraging large pretrained visual models in real-world applications constrained by limited resources, supported by an extensive experimental analysis. Our work shows that a simple, cost-efficient approach to supervised distillation from large pretrained models consistently achieves superior results. (ii) We investigate various data augmentation strategies based on Stable Diffusion and demonstrate that a variation of Mixup is notably efficient for distillation. Originally proposed by Pinkney (2022) in a different context to generate visually appealing combinations of images, it proves particularly effective when employed as a data augmentation technique for distillation. It operates

solely on unlabeled images, eliminating the necessity for text prompt engineering, and remains agnostic to the downstream task.

Concretely, our work reaches a series of experimental conclusions that ground our guidelines. Our experiments are conducted using DINOv2 teachers (Oquab et al., 2024), recognized for providing strong baselines (see Sec. 4.4.2) and extended to EVA-02 MIM- and CLIP-pretrained models (Fang et al., 2023; Sun et al., 2023) (see Appendix B.2.3). Our findings, summarized below, are validated across different architectures and various tasks: classification on specific image modalities, fine-grained classification, and semantic segmentation.

1. *Probing can yield better teachers than finetuning.* The remarkable adaptability of recent large-scale pretrained models, such as DINOv2, challenges the need for finetuning the teacher, which is standard practice in prior research on task-specific distillation (Jiao et al., 2020; Sun et al., 2019; Touvron et al., 2021; Beyer et al., 2022; Huang et al., 2023).

2. *Task-specific distillation complements task-agnostic distillation.* Task-specific distillation allows transferring task-specific knowledge, leading to better representations compared to simply finetuning the student after task-agnostic distillation, as illustrated in Fig. 4.1. We show that task-specific distillation consistently outperforms simple finetuning, which aligns with conclusions from prior works (Jiao et al., 2020; Huang et al., 2023), drawn for teachers finetuned on the target task. Our study extends their results to teachers that are only probed for the task, thus reducing the cost of the distillation procedure.

3. *Teachers do not need to be as accurate as their students.* This observation generalizes conclusions from early works (Yuan et al., 2020; Furlanello et al., 2018), conducted with teacher/student CNN models trained from scratch in a supervised manner. We show that even when DINOv2's pretrained ViT-S outperforms its teacher with simple finetuning, distillation can still be beneficial.

4. *Small models can directly learn from much larger ones.* Prior works suggest that a large capacity gap between teacher and student hinders distillation, and employ a middle-sized 'teacher assistant' to learn from the large model and teach the small one (Jiao et al., 2020; Mirzadeh et al., 2020; Wang et al., 2020). However, DINOv2's ViT-S was directly distilled from their ViT-g and yet demonstrates excellent generalization capabilities. Similarly, we show that task-specific distillation works equally well when using DINOv2's ViT-g or their middle-sized ViT-L to teach ViT-S.

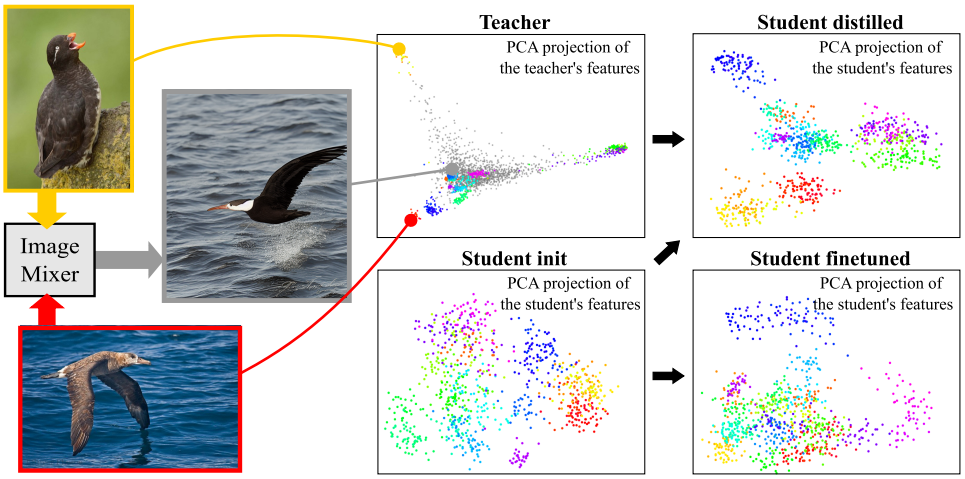


FIGURE 4.1: This paper advocates for distilling a large pretrained teacher (top, left) to train a small task-specific student model (top, right). This distillation process results in a better clustering of the representations compared to simply finetuning the student on the task (bottom, right). Distillation is improved by a class-agnostic data augmentation based on Stable Diffusion that consists in mixing real images to create synthetic ones, producing features shown in gray in the teacher plot. Each plot shows image features for 30 classes of the CUB Bird dataset, after PCA (one color per class).

5. *Diffusion models can be effectively leveraged as data augmentation for distillation without relying on class information*, making them applicable to tasks where text-conditioned image generation is non-trivial (such as semantic segmentation). To bypass the need for class information in Stable Diffusion, we leverage a diffusion model that generates *mixed* images conditionally on multiple images provided as input, taking inspiration from the classical Mixup augmentation. We show that, while being ineffective in the context of supervised learning, this mixing strategy consistently helps task-specific distillation.

4.2 Related work

In this section, we first discuss relevant prior work on knowledge distillation (Sec. 4.2.1). We then cover works that leverage Stable Diffusion for data augmentation, and discuss data augmentation in the context of distillation (Sec. 4.2.2).

4.2.1 Knowledge distillation

Task-specific vs. generic distillation. Following the pioneering work of Hinton et al. (2015), distillation has become a standard approach to transfer knowledge from one model into another (see Gou et al. 2021; Wang and Yoon 2021 for detailed surveys). Initially, knowledge distillation was conceived as a method to transfer knowledge from

a large teacher network trained on a specific task to a small student network (Hinton et al., 2015; Ba and Caruana, 2014). With the rise of self-supervised learning, the approach was extended to transfer general representations produced by a large generic model into small ones (Abbasi Koohpayegani et al., 2020; Fang et al., 2021; Xu et al., 2022; Gao et al., 2022; Navaneet et al., 2021; Wu et al., 2022; Duval et al., 2023). There, distillation is used as a knowledge compression mechanism, which is motivated by the observation that directly pretraining small models on large amounts of data leads to underwhelming results compared to learning them by distillation from large pretrained models (Abbasi Koohpayegani et al., 2020; Fang et al., 2021; Xu et al., 2022; Wu et al., 2022; Oquab et al., 2024).

In the context of self-supervised learning, it is then common to finetune distilled models on various downstream tasks, without further exploiting the teacher's knowledge (Sun et al., 2019; Touvron et al., 2021; Beyer et al., 2022). Surprisingly, only few studies have explored a task-specific distillation procedure that leverages both the teacher and the downstream task. An example is the two-stage distillation introduced in natural language processing by Jiao et al. (2020) and recently applied to vision tasks by Huang et al. (2023). Specifically, their approach involves a conventional generic distillation, followed by finetuning the teacher on a downstream task and applying a second task-specific distillation involving the finetuned teacher. In contrast, our findings indicate that finetuning the teacher is not always the optimal strategy, and we advocate for a less computationally demanding approach.

Architecture-dependent distillation. Some variants of knowledge distillation directly exploit the specific architecture of both the teacher and the student. These include feature-based knowledge distillation often tailored to CNNs (Romero et al., 2015; Zagoruyko and Komodakis, 2017; Chen et al., 2021a,b), where knowledge is distilled by matching representations from any intermediate layer(s), or aligning mutual relations in the feature space (Yim et al., 2017; Tung and Mori, 2019). Approaches specific to transformers have also emerged, consisting, for instance, of adding a separate distillation token (Touvron et al., 2021). Simultaneously, other works have proposed architecture-agnostic distillation approaches relying on particular loss functions (Tian et al., 2020; Zhao et al., 2022). For example, Tian et al. (2020) propose a contrastive objective inspired by self-supervised learning approaches. In our work, we adopt a task- and architecture-agnostic distillation framework, therefore bypassing the need for adjusting to the model's architecture.

4.2.2 Data augmentation

Traditionally, data augmentation has been used to improve the generalization capabilities of deep neural networks (Wang and Yoon, 2021). Recently, Stable Diffusion models have emerged as another compelling tool for data augmentation, and have been broadly studied in the context of supervised learning. In the context of knowledge distillation, data augmentation is not constrained by the need of class labels or segmentation masks, which suggests that optimal augmentation approaches may differ from those delineated in the context of supervised learning. Below we discuss prior works on the use of Stable Diffusion for data augmentation and prior studies on data augmentation for knowledge distillation.

Data augmentation with Stable Diffusion. Recent generative models such as latent diffusion models (Rombach et al., 2022) have emerged as a compelling way to artificially augment training data (Trabucco et al., 2023; Azizi et al., 2023; Dunlap et al., 2024) or even replace it (Sarıyıldız et al., 2023), usually using class names as textual prompts. Yet designing prompts can be difficult for tasks such as segmentation, as it requires featuring the multiple classes found in an image. Prior works often resort to prompt engineering (Fang et al., 2024) or to language models to generate prompts from class names (Nguyen et al., 2023; Zhou et al., 2022b).

An alternative to text-to-image generation is to leverage image-to-image diffusion models to directly provide training images as prompts. Image-to-image diffusion models have proven successful at various tasks such as restoration (Saharia et al., 2022a) or image editing (Brooks et al., 2023). However, using them as a tool for data augmentation raises significant challenges. These models can struggle with producing meaningful variations such as viewpoint changes or object shape variations, as pointed out by Brooks et al. (2023). Properties such as object shape, location, and appearance can be extracted and controlled from the internal representations of diffusion models (Epstein et al., 2023) but this requires manual interventions and cannot be universally applied to any task. For dense segmentation tasks, Yang et al. (2023) propose to generate synthetic data based on the segmentation mask of real images. This approach allows generating image/mask pairs without resorting to prompt engineering, but is restricted to supervised tasks with access to segmentation masks, and synthetic images are bound to be generated with these fixed masks. In contrast, we advocate an approach that can be universally applied to any task and that produces substantial image variations by interpolating between multiple training images (Pinkney, 2022).

Data augmentation in the context of distillation. In the context of knowledge distillation, Beyer et al. (2022) recommend to apply the same augmentations to the

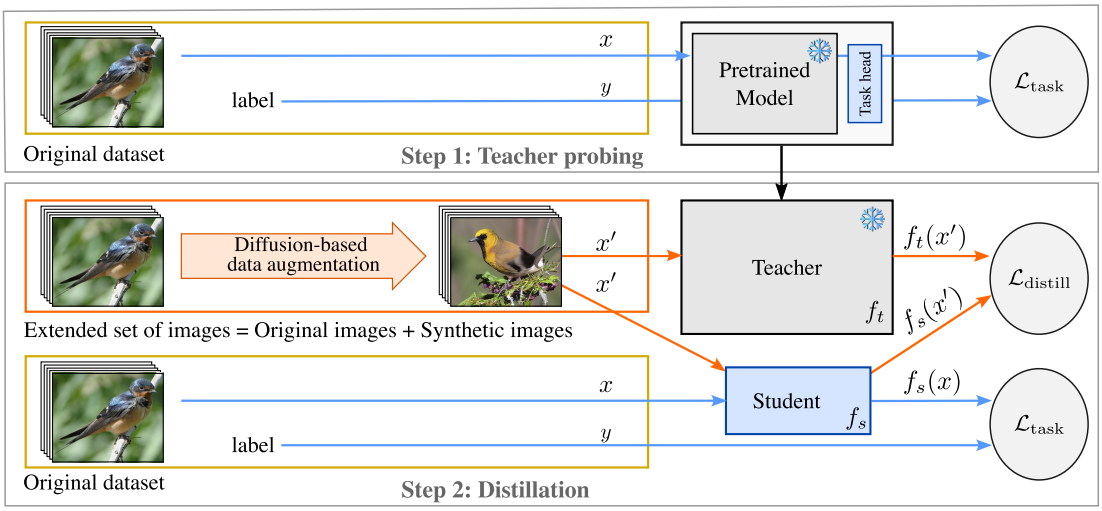


FIGURE 4.2: **Overview of the task-specific distillation pipeline.** The pretrained model is probed to build a teacher (top). Then its knowledge is distilled (Hinton et al., 2015) by minimizing the distillation loss $\mathcal{L}_{\text{distill}}$ jointly with the task loss $\mathcal{L}_{\text{task}}$ (bottom). $\mathcal{L}_{\text{distill}}$ is optimized with both i) original images x and ii) synthetic images obtained with Stable Diffusion x', while $\mathcal{L}_{\text{task}}$ is only optimized on the original dataset (x, y). Note that x and x' are also transformed using standard data augmentation (not shown here).

inputs of both teacher and student networks to ensure they are provided with consistent views. Wang et al. (2022) suggest that a good data augmentation scheme should reduce the covariance of the teacher-student cross-entropy, and propose an enhanced CutMix augmentation. Alternatively, Stanton et al. (2021) show the positive impact of Mixup on knowledge distillation. In this work, we show that distillation works better when performing data augmentation that goes beyond simple photometric and geometric transformations such as vanilla Mixup or CutMix, by exploiting the richness of generative models such as Stable Diffusion.

4.3 Method

Our study focuses on task-specific distillation, which consists in training a small model for a specific supervised task while transferring knowledge from a large pretrained encoder. In Sec. 4.3.1, we detail the standard approach for task-specific distillation of pretrained models. It usually divides in two steps: first training a teacher model on the target task, then transferring the knowledge from the trained teacher to a student. Unlike prior work, our distillation is performed without teacher finetuning: we only train a *task head* on the pretrained encoder. Finetuning can be computationally expensive, especially when dealing with large teachers such as ViT-g, but it may also compromise the quality of the visual representation acquired during pretraining (*e.g.*, from self-supervision). In Sec. 4.3.2, we present a mixing data augmentation based on

Stable Diffusion that leverages the teacher's knowledge more effectively to enhance the distillation process. The overall method is illustrated in Fig. 4.2.

4.3.1 Task-specific distillation

We consider a task, such as classification or segmentation, where the goal is to predict a label y (e.g. a class or a segmentation map) given an input image x. Typically, one could learn a model f to perform such a task using a training set $\mathcal{D}_{\text{train}}$ of image/labels pairs (x, y) by simply optimizing the following training loss (which is possibly regularized):

$$\mathcal{L}_{\text{task}}(f) = \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{train}}} \ell_{\text{task}}\left(f(x), y\right). \tag{4.1}$$

One can directly leverage a pretrained model by using it to initialize the model f, and then performing either *finetuning* or *probing* using objective $\mathcal{L}_{\text{task}}(f)$. However, these direct approaches require using the same architecture as the original pretrained model, which can be limiting for applications where inference speed and memory are critical factors. Instead, we are interested in learning a lightweight model f_s that can still leverage knowledge from a much larger pretrained encoder model e_t to perform the task. To this end, we first construct a teacher model f_t from the pretrained encoder e_t and then use it to distill knowledge relevant to the task on the lightweight model f_s.

Step 1: Teacher probing. We augment the encoder model e_t with a task-specific prediction head p_t creating the *teacher model* f_t (i.e., $f_t(x) = p_t(e_t(x))$). The teacher is then *probed* for the supervised task by training the prediction head p_t to minimize the training loss $\mathcal{L}_{\text{task}}(f_t)$. Notably, the parameters of the encoder e_t remain frozen. This not only significantly reduces the training cost compared to finetuning but it also helps preserving information acquired during (self-supervised) pretraining. Our experiments (Table 4.2) indeed show that this probed teacher leads to better distillation results than its finetuned version in general.

Step 2: Distillation. After probing the teacher f_t, we use it to guide the training of a smaller *student model* f_s on the downstream task. Specifically, we supplement the task loss $\mathcal{L}_{\text{task}}$ with a *distillation loss* $\mathcal{L}_{\text{distill}}$ that encourages the student's predictions to match the teachers', resulting in an overall objective of the form:

$$\mathcal{L}(f_s) := (1-\alpha)\mathcal{L}_{\text{task}}(f_s) + \alpha\mathcal{L}_{\text{distill}}(f_s, f_t), \tag{4.2}$$

where α is a weighting parameter controlling the strength of the distillation loss. We define $\mathcal{L}_{\text{distill}}$ as an average of some dissimilarity measure ℓ_{d} between the student's and

the teacher's predictions over a set $\mathcal{D}$ of *well-chosen* images:

$$\mathcal{L}_{\text{distill}}(f_s, f_t) = \mathbb{E}_{(x,.)\sim\mathcal{D}}\big[\ell_{\text{d}}\left(f_s(x), f_t(x)\right)\big]. \tag{4.3}$$

The loss in Eq. (4.3) ensures our distillation protocol is agnostic to the architecture since the dissimilarity measure ℓ_{d} depends solely on the student's and the teacher's outputs and not on their internal structure. We set the dissimilarity measure ℓ_{d} to be the KL-divergence rescaled by a temperature parameter T as proposed by Hinton et al. (2015):

$$\ell_{\text{d}}(f_s(x), f_t(x)) = T^2 D_{\text{KL}}\left(\frac{f_s(x)}{T}\middle\|\frac{f_t(x)}{T}\right). \tag{4.4}$$

The choice of images $\mathcal{D}$ in the distillation loss (Eq. (4.3)) is crucial as it defines the nature of images for which the student is required to match the teacher's predictions. While it is only natural to define $\mathcal{D}$ as the set of training images $\mathcal{D}_{\text{train}}$, this choice is not necessarily the most effective for extracting relevant knowledge from the teacher as $\mathcal{D}_{\text{train}}$ could offer a view that is too *narrow*. Instead, we propose to build $\mathcal{D}$ by extending $\mathcal{D}_{\text{train}}$ using an augmentation protocol based on Stable Diffusion, described in the next subsection.

4.3.2 Distilling with synthetic data

The distillation process outlined in Sec. 4.3.1 aims to align the teacher's and student's outputs for a set of images $\mathcal{D}$ that is sufficiently large and diverse to extract relevant knowledge. While it is possible to augment a dataset with standard data augmentation, our experiments indicate that this may not introduce enough diversity. When aiming for increased diversity, generating images relevant to the task—with suitable semantics and originating from the correct domain—is crucial. However, this step should be task-agnostic to avoid the need for manual tailoring to each downstream task, or to avoid providing class names or any other ground truth.

We propose to use a variant of Stable Diffusion, originally introduced by Pinkney (2022) for aesthetic purposes, named ImageMixer. It is a finetuned version of Rombach et al. (2022) that enables the *mixing* of CLIP image representations from two or more input images to generate a new one. More precisely, CLIP embeddings are concatenated along the sequence dimension and serve as a conditional input. We use this method as a variant of Mixup for data augmentation, which involves mixing random pairs of images, regardless of their classes. This enables us to create an augmented dataset $\mathcal{D}_{\text{sd}}$ containing both the original images from $\mathcal{D}_{\text{train}}$ and the synthetic ones. During training, we use $\mathcal{D}_{\text{sd}}$ by randomly sampling synthetic images and original ones with equal frequency.

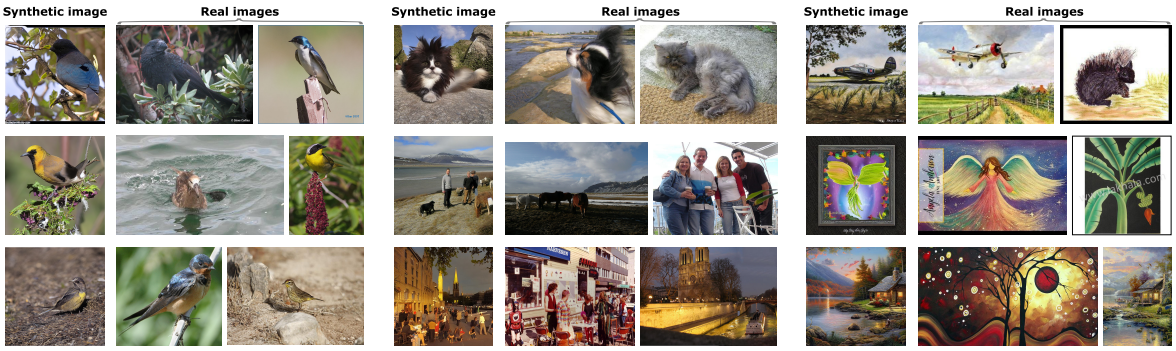


FIGURE 4.3: **Diffusion-based data augmentation**. Examples of synthetic images generated using ImageMixer (Pinkney, 2022) as described in Sec. 4.3.2, mixing two training images from CUB (Wah et al., 2011) (left), Pascal VOC (Everingham et al., 2010) (middle) and Painting from DomainNet (Peng et al., 2019) (right). Those populate the extended dataset $\mathcal{D}_{\text{sd}}$ for distillation.

Example images generated for the CUB (Wah et al., 2011), Pascal VOC (Everingham et al., 2010) and DomainNet's Painting (Peng et al., 2019) datasets can be found in Fig. 4.3. Additional examples can be found in the appendix.

It is crucial to note that the corresponding augmented set is exclusively used for the distillation loss $\mathcal{L}_{\text{distill}}$. We have experimentally observed that introducing synthetic data in the optimization of $\mathcal{L}_{\text{task}}$ degrades performance, even for a variant that only mixes images of the same class (see Sec. 4.4.3). This supports the intuition that the generated images are diverse enough to potentially extend beyond the scope of each class, while remaining close enough to the overall training domain to still be useful for distillation.

4.4 Experiments

We evaluate our distillation protocol across three families of tasks: classification on various domains, fine-grained classification, and semantic segmentation. For classification, we consider the painting, sketch and clipart datasets from DomainNet (Peng et al., 2019), each composed of the same 345 classes, for which we isolate 20% of the training set for testing. Fine-grained classification is conducted on the CUB (Wah et al., 2011), FGVC Aircraft (Maji et al., 2013) and DTD (Cimpoi et al., 2014) datasets respectively consisting of 200 bird species, 100 aircraft models, and 47 textures. Finally, we use three benchmarks for segmentation: ADE20K (Zhou et al., 2017), Cityscapes (Cordts et al., 2016), and the augmented Pascal VOC (Everingham et al., 2010). After an overview of our experimental setup (Sec. 4.4.1), we present our main distillation results (Sec. 4.4.2) followed by additional ablation studies (Sec. 4.4.3).

4.4.1 Experimental setting

We present the design choices for our student and teacher models and detail the data augmentation applied, the training hyperparameters and our evaluation protocol.

Backbone models. For the teacher, we start from one of the pretrained models provided by DINOv2 (Oquab et al., 2024), either ViT-S, ViT-L or ViT-g, three architectures of increasing capacity. Note that the ViT-L and ViT-S models provided by DINOv2 are distilled from their ViT-g. The teacher is then one of these pretrained models probed for the target downstream task (see top part of Fig. 4.2). We also consider a finetuned ViT-L teacher in our study to investigate the impact of finetuning versus probing strategies for the teacher. We do not explore finetuning the ViT-g model, in line with the paper's focus on maximizing the utility of pretrained models within constraints of limited computational resources.

For the student, we explore two lightweight architectures. The majority of our experiments use a ViT-S model initialized with DINOv2's pretrained weights. We also show that our observations generalize to randomly initialized models: we report experiments with a ResNet-50 model for classification and a DeepLabv3 model (Chen et al., 2017) with ResNet-50 backbone for segmentation.

Prediction head. We use a MLP head for classification (unlike DINOv2 which evaluates with a linear head) and DINOv2's linear head for segmentation, for students and for teachers. Note that there is no prediction head for the ResNet-50 and DeepLabv3 models as those are trained from scratch.

When available, the input for the prediction head is defined as follows. In classification tasks, we adhere to DINOv2's process: i) we concatenate the CLS tokens from up to the last four blocks (choosing 4 for DomainNet and 3 for fine-grained tasks), ii) optionally, we concatenate the average pooling of the patch embeddings from the last block (which we only do for DomainNet). For segmentation tasks, we adopt DINOv2's linear evaluation protocol, directly evaluating from the patch embeddings of the last block.

Synthetic image generation with Stable Diffusion. We generate synthetic datasets with n times more images than the original training set, setting n to 5 for DomainNet and segmentation tasks, and 10 for the relatively smaller fine-grained classification datasets. As noted earlier, this data augmentation strategy may be viewed as a variant of Mixup (Zhang et al., 2018), akin to interpolating between random pairs of images using the ImageMixer method proposed by Pinkney (2022).

Standard data augmentation. In all experiments, we apply classical data augmentation to both the original training images and the synthetic images. For classification tasks involving transformers, we use RandomResizedCrop, ColorJitter, and Mixup, while for ResNet-50, we use TrivialAugment (Müller and Hutter, 2021). Note that Mixup is excluded for synthetic images obtained from ImageMixer, which is already a variant of Mixup based on Stable Diffusion. For segmentation tasks, we adopt the same augmentations as DINOv2 (Oquab et al., 2024)—see details in the appendix. Following the recommendation of Beyer et al. (2022), the student and teacher models receive exactly the same batch of images, transformed with the same data augmentation.

Training hyperparameters. Probing runs for 20 epochs for ViT-L/g and 30 epochs for ViT-S, while finetuning lasts for 50 epochs for ViT-L and 80 epochs for ViT-S. We use the AdamW optimizer for training ViTs and SGD with momentum for ResNet-50, and a cosine scheduler in both cases. The selection of weight decay and learning rate is determined through a grid search on the validation set, with specific details available in the appendix. In instances where no predefined validation set exists, we allocate 10% of the training set for this purpose. We use a fixed distillation temperature of $T = 2$ and a constant weighting between $\mathcal{L}_{\text{task}}$ and $\mathcal{L}_{\text{distill}}$ set to $\alpha = 0.5$ for all experiments.

Evaluation. We report results averaged over three independent runs with different random seeds. For distillation evaluations, we consider three different teachers, each from independent runs, and conduct 2 runs per teacher for DomainNet and 3 runs for fine-grained and segmentation tasks.

About our probing results. We remind that for classification, we use a MLP head while DINOv2 (Oquab et al., 2024) uses a linear head. Please also note that for segmentation, we use an image size of 560×560 pixels while DINOv2 uses 512 × 512. This explains why our probing results are slightly higher than those reported by Oquab et al. (2024) (comparison in the appendix).

4.4.2 Experimental results

We explore distillation with two different students: i) DINOv2's ViT-S pretrained with task-agnostic distillation, and ii) randomly initialized models: a ResNet-50 for classification and a DeepLabv3 with ResNet-50 backbone for segmentation.

Our results are presented in Tables 4.1 to 4.3. Table 4.1 reports probing and finetuning results for DINOv2 ViT-S, ViT-L and ViT-g pretrained models. Table 4.2 reports distillation results using ViT-S as the student, and using a probed ViT-S, a probed and a finetuned ViT-L or a probed VIT-g as the teacher. Table 4.3 reports distillation

Model		Classification on DomainNet (acc)			Fine-grained classification (acc)			Semantic segmentation (mIoU)		
		Painting	Sketch	Clipart	CUB	Aircraft	DTD	ADE20K	Cityscapes	VOC
ViT-S	*(1a)* Probing	77.3	71.9	79.3	88.2	77.1	82.1	45.1	67.0	81.8
	(1b) Finetuning	79.4	76.0	81.8	87.3	87.8	81.6	49.8	75.8	84.6
ViT-L	*(2a)* Probing	82.9	80.4	85.3	91.3	87.8	85.5	47.8	70.4	82.7
	(2b) Finetuning	83.9	81.4	85.9	91.5	94.0	85.8	57.4	78.6	88.0
ViT-g	*(3a)* Probing	83.0	81.2	85.7	91.6	88.1	85.8	48.8	71.2	83.5

TABLE 4.1: **Probing/finetuning of DINOv2 pretrained models** for classification on DomainNet, fine-grained classification and semantic segmentation. We report accuracy for classification and mIoU for segmentation. Relative distillation gains in Table 4.2 are with respect to underlined results in this table.

Student	Teacher		SD	Classification on DomainNet (acc)			Fine-grained classification (acc)			Semantic segmentation (mIoU)		
				Painting	Sketch	Clipart	CUB	Aircraft	DTD	ADE20K	Cityscapes	VOC
ViT-S	ViT-S probed	*(4a)*	✗	80.0 (+0.6)	76.9 (+0.9)	82.2 (+0.4)	89.4 (+1.7)	86.5 (-1.3)	82.9 (+0.8)	49.6 (-0.2)	71.2 (-4.6)	84.6 (+0.0)
		(4b)	✓	80.2 (+0.8)	77.1 (+0.2)	82.4 (+0.6)	89.7 (+1.5)	86.6 (-1.2)	83.4 (+1.3)	50.3 (+0.5)	72.3 (-3.5)	84.9 (+0.3)
	ViT-L probed	*(5a)*	✗	80.5 (+1.1)	77.8 (+1.8)	**83.4** (+1.6)	89.7 (+1.5)	89.2 (+1.4)	83.4 (+1.3)	50.7 (+0.9)	74.0 (-1.8)	85.5 (+0.9)
		(5b)	✓	**80.8** (+1.4)	**78.0** (+2.0)	83.2 (+1.4)	**90.0** (+1.8)	**89.8** (+2.0)	**84.0** (+1.9)	**51.7** (+1.9)	74.7 (-1.1)	**86.1** (+1.5)
	ViT-L finetuned	*(6a)*	✗	79.7 (+0.3)	77.0 (+1.0)	82.5 (+0.7)	88.6 (+0.4)	88.9 (+1.3)	81.5 (-0.6)	50.7 (+0.9)	**76.3** (+0.5)	84.8 (+0.2)
		(6b)	✓	80.3 (+0.9)	77.2 (+1.2)	82.9 (+1.1)	88.6 (+0.4)	89.1 (+1.5)	82.5 (+0.4)	51.6 (+1.8)	**76.4** (+0.6)	85.7 (+1.1)
	ViT-g probed	*(7a)*	✗	80.5 (+1.1)	77.7 (+1.7)	**83.4** (+1.6)	89.1 (+0.9)	**89.6** (+1.8)	83.1 (+1.0)	51.6 (+1.8)	74.4 (-1.4)	85.7 (+1.1)
		(7b)	✓	**80.8** (+1.4)	**78.0** (+2.0)	**83.3** (+1.5)	**89.8** (+1.6)	**90.1** (+2.3)	**83.6** (+1.5)	**52.1** (+2.3)	75.0 (-0.8)	**86.3** (+1.7)

TABLE 4.2: **Distillation on ViT-S initialized with DINOv2** for classification on DomainNet, fine-grained classification and semantic segmentation. We report accuracy for classification and mIoU for segmentation. We report results with and without data augmentation based on Stable Diffusion (SD), for various choices of teachers. Relative gains with respect to simple probing or finetuning (best underlined in Table 4.1) are in parentheses. **Bold** numbers: within 95% confidence interval of the best score for each task.

results using a probed ViT-g as the teacher, and a randomly initialized ResNet-50 as the student. Distillation results reported in Tables 4.2 and 4.3, both with and without augmenting the training set with synthetic images for distillation, are compared to those obtained with a simple probing or finetuning of the ViT-S (best underlined in Table 4.1) and with simple training of ResNet-50 (underlined in Table 4.3), with relative gains indicated in parentheses.

We discuss the results of Tables 4.1 to 4.3 according to four separate axes supporting the main claims of this study: i) the relative gains of distillation over finetuning when teaching a small pretrained model, ii) the impact of finetuning the teacher, iii) the impact of using a teacher that is less accurate than the student, and iv) the generalization of our observations to students trained from scratch.

In what follows, observations are discussed by comparing lines of Tables 4.1 to 4.3. The lines from these three tables are denoted by unique alphanumerical reference such that *e.g.* the mention *(2a vs 2b)* refers to comparing lines *2a* and *2b* in the Table 4.1.

Task-specific distillation complements task-agnostic distillation. The key observation from Table 4.2 is that *task-specific distillation generally outperforms probing*

Student	Teacher		SD	Classification on DomainNet (acc)			Fine-grained classification (acc)			Semantic segmentation (mIoU)		
				Painting	Sketch	Clipart	CUB	Aircraft	DTD	ADE20K	Cityscapes	VOC
R50	-	*(8a)*	-	66.0	68.1	72.5	73.3	85.0	63.5	37.8	**67.9**	67.5
	ViT-g probed	*(9a)*	✗	67.7 (+1.3)	70.5 (+2.4)	**74.9** (+2.4)	76.0 (+2.7)	85.7 (+0.7)	66.7 (+3.2)	38.2 (+0.4)	**67.7** (-0.2)	67.7 (+0.2)
		(9b)	✓	**69.1** (+2.7)	**71.0** (+2.9)	**75.2** (+2.7)	**79.1** (+5.8)	**87.8** (+2.8)	**69.4** (+5.9)	**42.1** (+4.3)	**69.3** (+1.4)	**73.9** (+6.2)

TABLE 4.3: **Distillation from ViT-g to ResNet-50 (resp. DeepLabv3-ResNet50 for segmentation) trained from scratch** for classification on DomainNet, fine-grained classification and semantic segmentation. We report accuracy for classification and mIoU for segmentation. We report results with and without data augmentation based on Stable Diffusion (SD). Relative gains with respect to simple training (underlined) are in parentheses. **Bold** numbers: within 95% confidence interval of the best score for each task.

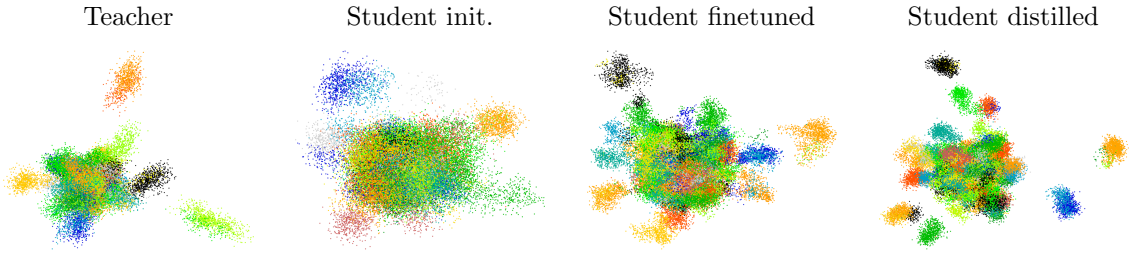


FIGURE 4.4: PCA of patch embedding representations for 20 classes of ADE20K for the ViT-g teacher (a) and for the ViT-S student in its initial state (b), after finetuning (c) and after distillation (d), colored by their main class (details in the appendix). Classes are better clustered after distillation than after finetuning.

and finetuning. This can be observed by comparing any line from Table 4.2, *4* to *7*, *a* or *b*, with the corresponding number in line *1* from Table 4.1, referred to as *(4-7 vs 1)* following the notation introduced in the previous paragraph. Fig. 4.4 illustrates this observation on ADE20K: the PCA of patch embedding representations exhibits a better clustering structure after distillation than after finetuning (see also Fig. 4.1). Cityscapes is the only exception where distillation from a probed ViT-L or a probed ViT-g does not improve over finetuning. Interestingly, on Cityscapes, the finetuned ViT-S student already outperforms the probed ViT-g and ViT-L teachers by a large margin (+4.6 and +5.4 mIoU) *(1b vs 3a,2a)*, which may explain why distilling from those is not beneficial.

Our dataset augmentation based on Stable Diffusion further enhances distillation results (*4a vs 4b, 5a vs 5b*, etc.), except on Clipart, where it performs on par with distillation on the original training images alone.

While ViT-g exhibits slightly higher accuracy than ViT-L when probed on downstream tasks *(3a vs 2a)*, both models serve as almost equally effective teachers for distillation *(7 vs 5)*. In exploring experiments with smaller teachers, we evaluate distillation from a probed ViT-S *(1a)*, placing ourselves in the context of self-distillation. We observe

that when the performance gap between probing and finetuning is not too large, self-distillation *(4)* improves over finetuning *(1b)*, evident across all baselines except for Aircraft and Cityscapes, where the probed ViT-S has an accuracy/mIoU approximately 10% lower than with finetuning *(1a vs 1b)*. However, it is important to note that ViT-g and ViT-L remain superior teachers compared to ViT-S. This implies that, even if ViT-S was pretrained with generic distillation from ViT-g, *it is more effective to directly leverage the largest teachers for downstream tasks.*

Finetuning yields a poorer teacher than probing. Next, we study the impact of finetuning our teacher prior to distillation, comparing distillation results using either a probed or finetuned pretrained ViT-L model from DINOv2 (Oquab et al., 2024). Finetuning significantly enhances ViT-L's accuracy compared to probing *(2a vs 2b)*. However, employing the finetuned ViT-L model as a teacher generally results in a poorer performance for the student *(5 vs 6)*. For example, finetuning brings about 6% increase in accuracy for Aircraft and Pascal VOC compared to probing *(2a vs 2b)*, yet the distillation results with the probed teacher are better *(5 vs 6)*. This suggests that *preserving the rich representations learned during pretraining is crucial, even if it leads to a teacher with lower accuracy for the specific task.* Cityscapes is the only exception where distillation of a finetuned teacher significantly improves results, while using a probed teacher degrades them *(5 vs 6)*. This may be attributed to the substantial performance gap between probing and finetuning on this dataset, with +8 to +9 in mIoU *(1a vs 1b, 2a vs 2b)*.

In summary, our experiments indicate that *finetuning the teacher for task-specific distillation is often unnecessary and sometimes even detrimental.* Given the relatively fast training of the MLP head, the primary computational cost in knowledge distillation with teacher probing lies in training the student, with the additional overhead of performing forward passes through the teacher.

Teachers are not required to be as accurate as students. Sometimes, simply finetuning our ViT-S model *(1b)* gives better results than probing ViT-g *(3a)*. This is the case for all three segmentation tasks. Still, distilling from ViT-g proves beneficial for ADE20K and Pascal VOC *(7b)*, as it gives around 2% mIoU gain compared to finetuning *(1b)*, even though the finetuned ViT-S model is already about 1% higher in mIoU than the teacher *(3a)*. This supports the more general observation that *a student can surpass its teacher and still benefit from distillation.*

Data augmentation based on Stable Diffusion substantially helps students trained from scratch. Here, we replace DINOv2's pretrained ViT-S student with a ResNet-50 (resp. DeepLabv3 with a ResNet-50 backbone for segmentation) trained

from scratch, while retaining DINOv2's pretrained ViT-g as the teacher. Results are reported in Table 4.3 and consistently demonstrate that distillation is beneficial, leading to 2% accuracy gain on average. Notably, data augmentation based on Stable Diffusion significantly enhances results, yielding a further 2-3% accuracy gain on fine-grained tasks and a 4-6% mIoU gain for segmentation compared to standard distillation *(9a vs 9b)*. Surprisingly, the ResNet-50 model benefits even more from distillation than the pretrained ViT-S model. These findings indicate that the observations made for a pretrained ViT-S student generalize to students that i) did not undergo generic distillation or any form of pretraining, and ii) whose architecture is not based on transformers like the teacher.

4.4.3 Ablation studies

Data augmentation with Stable Diffusion. We now compare various strategies for creating an augmented dataset $\mathcal{D}_{\text{sd}}$ used for distillation. Our evaluation focuses on fine-grained classification tasks, involving both a pretrained ViT-S and a ResNet-50 trained from scratch as students.

We conduct a comparative analysis of our data augmentation strategy based on ImageMixer (Pinkney, 2022). We compare it to: i) another model by Pinkney (2022) creating image variations from single images; ii) an augmentation approach incorporating an ImageNet subset; and iii) the text-to-image diffusion model used by Sarıyıldız et al. (2023). For the latter, we explore textual prompts with the parent class, and with and without class names. Specifically, prompts with class names take the form "A photo of a {class name} {parent class}" while prompts without class names follow the pattern "A photo of a {parent class}", where {parent class} represents either bird, aircraft, or texture.

Table 4.4 shows that our prompt-free approach performs equally well, if not better, than a prompt-based data augmentation that leverages class information. Additionally, prompt engineering poses challenges for certain tasks, especially segmentation, and necessitates the model used for parsing prompts (*e.g.*, CLIP) to be trained with semantic information about the data. This may be impossible for some modalities such as medical or microscopy images, which are not easily described with text and might fall outside the semantic scope expected by CLIP-like models.

Our chosen strategy based on synthetic images produced using the ImageMixer model outperforms the ImageVariations model on 5 out of 6 settings, validating the benefits of a mixing-based approach conditioning image generation with multiple images.

	Text prompt-free	Student: ViT-S			Student: ResNet-50		
		CUB	Aircraft	DTD	CUB	Aircraft	DTD
Baseline (distillation only from $\mathcal{D}_{\text{train}}$)	✓	89.1	89.6	83.1	76.0	85.7	66.7
$\mathcal{D}_{\text{sd}}$ uses Text-to-image (parent class)	✗	89.4	90.0	83.6	77.8	87.9	68.1
$\mathcal{D}_{\text{sd}}$ uses Text-to-image (class name)	✗	89.5	90.2	83.5	79.8	87.6	68.7
$\mathcal{D}_{\text{sd}}$ composed of Random ImageNet images	✓	89.2	89.4	83.3	77.3	86.5	65.8
$\mathcal{D}_{\text{sd}}$ uses ImageVariations	✓	89.5	90.4	83.4	78.8	87.7	68.7
$\mathcal{D}_{\text{sd}}$ uses **ImageMixer**	✓	89.8	90.1	83.6	79.1	87.8	69.4

TABLE 4.4: **Building $\mathcal{D}_{\text{sd}}$ from $\mathcal{D}_{\text{train}}$.** We compare distillation results, using ViT-g as a teacher and ViT-S or ResNet-50 as a student, for our mixing approach based on stable diffusion (ImageMixer, by Pinkney 2022) with i) a model producing image variations from single images (ImageVariations, by Pinkney 2022), ii) simply adding a subset of ImageNet, and iii) text-to-image diffusion using the parent class only, *i.e.* bird, aircraft or texture ("A photo of a {parent class}"), or using class information as well ("A photo of a {class name} {parent class}"). We observe that the ImageMixer variant we advocate for is surprisingly competitive despite not requiring a text prompt.

	Data used in .. $\mathcal{L}_{\text{distill}}$	$\mathcal{L}_{\text{task}}$	CUB	Aircraft	DTD
Finetuning	-	$\mathcal{D}_{\text{sd-intra}}$	83.8	85.4	80.3
		$\mathcal{D}_{\text{train}}$	87.3	87.8	81.6
Distillation	$\mathcal{D}_{\text{sd-intra}}$	$\mathcal{D}_{\text{sd-intra}}$	89.6	89.0	83.6
	$\mathcal{D}_{\text{sd-intra}}$	$\mathcal{D}_{\text{train}}$	89.6	90.1	83.9
	$\mathcal{D}_{\text{sd}}$	$\mathcal{D}_{\text{train}}$	89.8	90.1	83.6

TABLE 4.5: **Impact of synthetic data on each loss.** Impact on fine-grained classification tasks for finetuning and distillation with ViT-S as student and ViT-g as teacher, using a dataset $\mathcal{D}_{\text{sd-intra}}$ augmented with synthetic images by mixing original images inside each class separately.

	Loss	SD	CUB	Aircraft	DTD
Finetuning	$\mathcal{L}_{\text{train}}$	-	87.3	87.8	81.6
Distillation	$\mathcal{L}_{\text{distill}}$	✗	88.8	86.2	82.7
		✓	89.6	86.9	82.9
	$\mathcal{L}_{\text{train}}+\mathcal{L}_{\text{distill}}$	✗	89.1	89.6	83.1
		✓	89.8	90.1	83.6

TABLE 4.6: **Role of the different losses**. We compare optimizing $\mathcal{L}_{\text{train}}$ only (*i.e.* finetuning), $\mathcal{L}_{\text{distill}}$ only, or both losses with equal weighting (standard distillation followed in our experiments). Results are with ViT-S as student and ViT-g as teacher.

Using augmented data for supervision. In our study, we use synthetic data only for optimizing the distillation loss $\mathcal{L}_{\text{distill}}$, while the task-specific loss $\mathcal{L}_{\text{task}}$ is trained solely on real data. In this section, we explore the outcomes when incorporating synthetic images as additional labeled data for optimizing $\mathcal{L}_{\text{train}}$, for both finetuning and distillation. For this purpose, we compare two different ways of leveraging the difusion model of Pinkney (2022) described in Sec. 4.3: mixing images regardless of their labels (inter-class), or mixing images from each class separately (intra-class). These approaches result in augmented datasets $\mathcal{D}_{\text{sd}}$ and $\mathcal{D}_{\text{sd-intra}}$ containing both the original images and the synthetic ones, as explained in Sec. 4.3. Table 4.5 presents distillation results for the fine-grained datasets. We observe comparable performance between inter-class and intra-class approaches, but incorporating synthetic data for supervision is not beneficial. In particular, including synthetic images for finetuning considerably degrades results. This aligns with the intuition that the diffusion model may not be faithful enough to each fine-grained class, and even when provided with two images of

the same class, it may generate a new image beyond the scope of this class.

Relative weighting between task and distillation losses. Here, we investigate the influence of completely excluding the loss $\mathcal{L}_{\text{task}}$ during distillation. Table 4.6 presents the results for fine-grained classification when solely optimizing $\mathcal{L}_{\text{train}}$ (*i.e.*, finetuning), solely optimizing $\mathcal{L}_{\text{distill}}$, and optimizing both with equal weighs, as implemented in our study. The outcomes reveal that training without label information (*i.e.*, optimizing $\mathcal{L}_{\text{distill}}$ only) yields competitive results for CUB but significantly lower results for Aircraft. Overall, the student achieves the best results when exposed to both hard labels and soft teacher labels.

4.5 Discussion and concluding remarks

Since the seminal work of Sharif Razavian et al. (2014), it has been known that generic pretrained models could be reused directly or adapted for many target tasks instead of learning new models from scratch. Yet, the rapid development and public release of even larger, rich and generic models, pretrained on up to billions of images (Oquab et al., 2024; Fang et al., 2023), raises a pressing question with heavy practical implications: *How to best leverage the knowledge of large and generic visual models when training a smaller model for a specific task*? Our work aims at addressing this question by reexamining current good practices for knowledge distillation in the light of these new large models, and draws a series of experimental conclusions. Below we summarize the main messages of our study, and relate them to previous discussions on neighboring topics.

Task-specific distillation and self-supervised learning. In the context of self-supervised learning, knowledge distillation has emerged as a compelling way to compress large pretrained models into smaller ones, yielding significant improvements compared to directly pretraining these small models (Abbasi Koohpayegani et al., 2020; Fang et al., 2021; Xu et al., 2022; Wu et al., 2022; Oquab et al., 2024). Yet, our study shows that simply finetuning or probing these small pretrained models yields sub-optimal results compared to leveraging the knowledge of the larger models for a specific downstream task.

Accurate teachers may not be the best for distillation. In prior works, Cho and Hariharan (2019); Mirzadeh et al. (2020) have observed that the most accurate teachers are not always the best for distillation, attributing this observation to the model capacity gap between the student and the teacher. To make the most of the largest

models, Mirzadeh et al. (2020) propose a multi-stage approach where knowledge is distilled from a large model to successively smaller ones, thus reducing the capacity gap between two successive distillation steps. Pointing to the inefficiency of such approach, Cho and Hariharan (2019) show that instead, this capacity gap can be mitigated by stopping the teacher's training early. This early-stopping approach may be related to our observation: by freezing the teacher's pretrained backbone, *i.e.*, *probing* the model instead of *finetuning* it, we prevent it from specializing too much to the given task. This results in better distillation despite a lower teacher accuracy. Nevertheless, we did not find any evidence that a large capacity gap may be detrimental to distillation, since our best results were obtained by distilling directly from DINOv2's (Oquab et al., 2024) largest ViT-g model to much smaller models, such as ViT-S or ResNet-50. Instead, the fact that a probed model can serve as a better teacher than a finetuned one suggests that some aspects of the representation that are still relevant to the task are lost when training for that task. This phenomenon is further discussed in the next paragraph.

Probing can yield better teachers than finetuning. One of the key observations of our study is that, at comparable performance (experimentally, a difference in accuracies smaller than 6% for the three families of models considered here, DINOv2, EVA-02 and EVA-02-CLIP), a probed model makes a better teacher than a finetuned one. This observation could be due to the *catastrophic forgetting* which happens (Kirkpatrick et al., 2017) when performing finetuning. Our experiments suggest that, when specializing for a given task, the finetuned teacher has forgotten some features that are not immediately relevant for optimizing the task loss, but that still help generalization. For instance, when finetuned on the CUB bird classification task, the pretrained model could end up only relying on *spurious correlations* (such as the background, a standard source of spurious correlations in CUB, as studied by Sagawa et al. 2019) that provide *shortcuts* to the optimization. These shortcuts help improving the teacher accuracy but are detrimental to the distillation process.

A teacher need not be more accurate than its student. Prior works have showed that student models can learn from poorly trained teachers (Yuan et al., 2020), or teachers with the same architecture (Furlanello et al., 2018), sometimes even outperforming them. Our results are consistent with these findings, since we also observed that a small model can benefit from distillation even when that same model already outperforms its larger teacher after simple finetuning. Additionally, distillation also resulted in improvements when using a teacher of the same size as the student (*i.e.*, self-distillation). However, our results highlight that distilling from the largest models

works considerably better than self-distillation, thus supporting the idea that knowledge from larger models can further guide the training of a smaller student.

Stable Diffusion as a source of additional information. Our study shows that our data augmentation strategy producing synthetic images with Stable Diffusion can be leveraged to extend the set of images used to optimize the distillation loss $\mathcal{L}_{\text{distill}}$. However, as illustrated in Table 4.5, including these synthetic images in the optimization of the task loss $\mathcal{L}_{\text{task}}$ can degrade results, particularly for finetuning. This shows that it may be difficult to efficiently leverage this augmentation strategy outside the context of knowledge distillation. Using synthetic image-label pairs for supervised learning requires i) the generation process to be good enough for images to pertain to their class, which may be challenging for fine-grained tasks, and ii) the generation process to create the right label for each new image, which is particularly challenging for dense tasks such as semantic segmentation. Knowledge distillation alleviates the need for generating class-specific images and for labeling generated images, and hence can more easily leverage Stable Diffusion.

Chapter 5

LUDVIG: Learning-free Uplifting of 2D Visual features to Gaussian Splatting scenes

We address the problem of extending the capabilities of vision foundation models such as DINO, SAM, and CLIP, to 3D tasks. Specifically, we introduce a novel method to uplift 2D image features into Gaussian Splatting representations of 3D scenes. Unlike traditional approaches that rely on minimizing a reconstruction loss, our method employs a simpler and more efficient feature aggregation technique, augmented by a graph diffusion mechanism. Graph diffusion refines 3D features, such as coarse segmentation masks, by leveraging 3D geometry and pairwise similarities induced by DINOv2. Our approach achieves performance comparable to the state of the art on multiple downstream tasks while delivering significant speed-ups. Notably, we obtain competitive segmentation results using generic DINOv2 features, despite DINOv2 not being trained on millions of annotated segmentation masks like SAM. When applied to CLIP features, our method demonstrates strong performance in open-vocabulary object localization tasks, highlighting the versatility of our approach.[1]

This work, co-authored with Romain Menegaux, Michael Arbel, Diane Larlus, and Julien Mairal, was published as an arXiv preprint [arXiv:2410.14462].

[1]Project page: https://juliettemarrie.github.io/ludvig

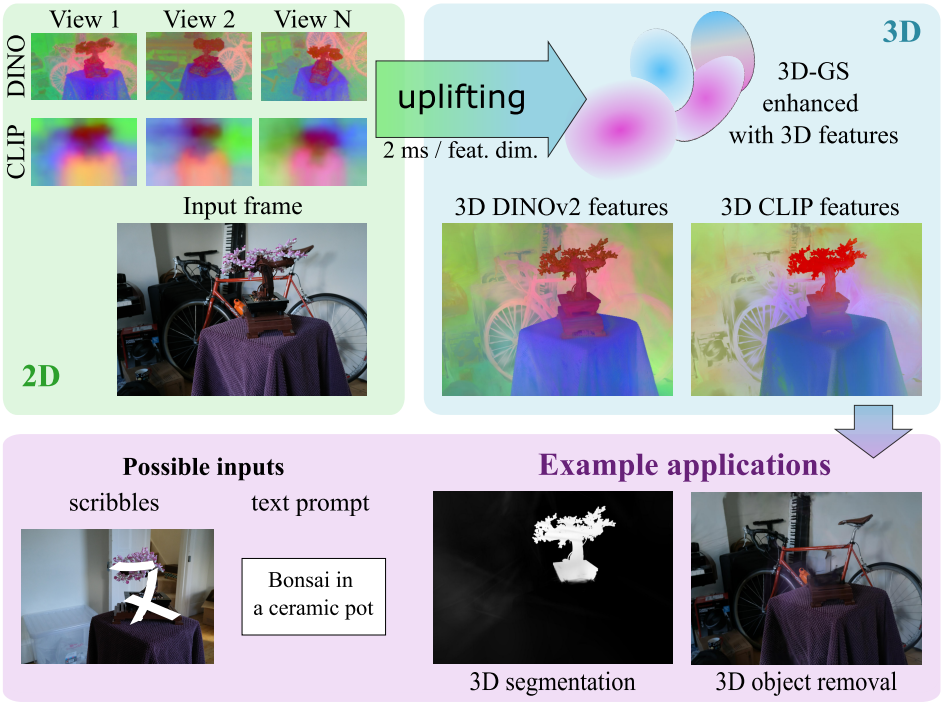


FIGURE 5.1: In this paper, we propose a simple, parameter-free method to uplift any 2D visual features (e.g., CLIP or DINO) into a 3D Gaussian Splatting (3D-GS) representation. Uplifting is directly implemented into the rendering process, and takes about 2ms per image and feature dimension. The uplifted features can be leveraged for various 3D tasks, such as segmentation or object removal, based on various inputs, such as scribbles or text prompts.

5.1 Introduction

The field of image understanding has made remarkable progress, driven by large pretrained models such as CLIP (Radford et al., 2021), DINO (Caron et al., 2021; Oquab et al., 2024), or SAM (Kirillov et al., 2023), also called foundation models. A key factor behind their exceptional generalization capabilities lies in the size of their training datasets, often composed of millions or even billions of samples.

Meanwhile, 3D scene representation has also advanced through machine learning approaches like NeRF (Mildenhall et al., 2021) and model fitting techniques such as Gaussian Splatting (Kerbl et al., 2023). These methods typically reconstruct scenes from a few dozen views captured from different angles, effectively preserving both appearance and geometric details. However, they are not suited for semantic tasks.

The complementarity of these two families of approaches has been exploited to integrate geometry and semantics by uplifting image-level features extracted by large pretrained models into NeRF or Gaussian Splatting 3D representations. This has led to a surge in methods for language-guided object retrieval (Kerr et al., 2023; Liu et al., 2023b; Zuo et al., 2024; Wu et al., 2024b), scene editing (Kobayashi et al., 2022; Chen et al., 2024; Fan et al., 2023), or semantic segmentation (Cen et al., 2023c; Ye et al., 2024; Ying et al., 2024).

The main limitation of these approaches lies in their dependence on optimization, which requires an iterative process to learn a scene-specific 3D representation by minimizing a reprojection error across all training views. While this loss function is intuitive, a faster and more straightforward method for transferring 2D generic visual features to *already trained* Gaussian Splatting 3D models would be preferable, which is one of the purposes of this work.

In this paper, we demonstrate that a simple, learning-free process is highly effective for uplifting 2D features or semantic masks into 3D Gaussian Splatting scenes. This process, which can be viewed as an 'inverse rendering' operation, is both computationally efficient and adaptable to any feature type. We showcase its effectiveness by uplifting visual features from DINOv2 (Oquab et al., 2024; Darcet et al., 2024), semantic masks from SAM (Kirillov et al., 2023) and SAM2 (Ravi et al., 2024), and visual features from CLIP (Ilharco et al., 2021), enabling language-driven applications. Then, we show that a graph diffusion mechanism (Kondor and Lafferty, 2002; Smola and Kondor, 2003) is helpful for feature refinement in 3D scenes. This mechanism is rooted in spectral graph theory and spectral clustering (Belkin and Niyogi, 2001; Shi and Malik, 2000; Meila and Shi, 2000). In the context of our work, it transforms coarse segmentation inputs, such as scribbles or alignment scores between visual features and a text query, into accurate 3D segmentation masks without the need for segmentation models such as SAM. When evaluated on segmentation and open-vocabulary object localization tasks, our method matches state-of-the-art performance while being significantly faster than previous optimization-based approaches.

To summarize, our contributions are threefold: (i) We introduce a simple, learning-free uplifting approach that can be directly integrated into the rendering process, achieving state-of-the-art results when applied to SAM-generated semantic masks (Sec. 5.4.1); (ii) We demonstrate that using graph diffusion based on uplifted DINOv2 features yields competitive results for foreground/background and open-vocabulary object segmentation, despite DINOv2 not being explicitly trained for segmentation like SAM (Secs. 5.4.1, 5.4.2); (iii) We show that combining SAM with graph diffusion achieves state-of-the-art results on open-vocabulary object segmentation tasks (Sec. 5.4.2).

5.2 Related work

Learning 3D semantic scene representations with NeRF. NeRF (Mildenhall et al., 2021) uses a multilayer perceptron to predict the volume density and radiance for any given 3D position and viewing direction. Such a representation can naturally be extended to semantic features. The early works N3F (Tschernezki et al., 2022)

and DFF (Kobayashi et al., 2022) distill DINO 2D (*i.e.*, image-level) features (Caron et al., 2021) in scene-specific NeRF representations. Kobayashi et al. (2022) also distill LSeg (Li et al., 2022a), a language-driven model for semantic segmentation. Building on this, LERF (Kerr et al., 2023) and 3D-OVS (Liu et al., 2023b) learn 3D CLIP (Radford et al., 2021) and DINO (Caron et al., 2021) features jointly for open-vocabulary segmentation. These works were extended to other pretrained models such as latent diffusion models (Ye et al., 2023) or SAM (Kirillov et al., 2023) for segmentation (Cen et al., 2023c; Ying et al., 2024).

Learning 3D semantic scene representations with GS. Subsequent work has relied on the Gaussian Splatting method (Kerbl et al., 2023), enabling rendering speeds orders of magnitude faster than NeRF-based models. Several tasks have been addressed such as semantic segmentation using SAM (Cen et al., 2023b; Ye et al., 2024; Kim et al., 2024; Wu et al., 2024b), language-driven retrieval or editing using CLIP combined with DINO (Zuo et al., 2024) or SAM (Ye et al., 2023; Wu et al., 2024b), scene editing using diffusion models (Chen et al., 2024; Wang et al., 2024a), and 3D-aware finetuning (Yue et al., 2024). These works learn 3D semantic representations by minimizing a reprojection loss. As a single scene can be represented by over a million Gaussians, such optimization-based techniques have strong memory and computational limitations. To handle these, FMGS (Zuo et al., 2024) employs a multi-resolution hash embedding (MHE) of the scene for uplifting DINO and CLIP representations, Feature 3DGS (Zhou et al., 2024) learns a 1×1 convolutional upsampler of Gaussians' features distilled from LSeg and SAM's encoder, and LangSplat (Qin et al., 2024) learns an autoencoder to reduce CLIP feature dimension from 512 to 3. In contrast, our approach requires no learning, which significantly speeds up the uplifting process and reduces the memory requirements.

Direct uplifting of 2D features into 3D. Direct uplifting from 2D to 3D has been explored in prior works. GaussianEditor (Chen et al., 2024) uplifts 2D SAM masks into 3D to selectively optimize Gaussians for editing tasks, see details in the supplementary (Sec. C.3.3). Dr. Splat (Jun-Seong et al., 2025) also lifts 2D SAM masks, to create an assignment from object CLIP features to Gaussians. Both approaches are specific to segmentation and are ill-suited for uplifting generic representations. Semantic Gaussians (Guo et al., 2024) pairs 2D pixels with 3D Gaussians along each pixel's ray based on depth information but relies on a learned 3D convolutional network. OpenGaussian (Wu et al., 2024b) also pairs pixels with Gaussians, but relies on pseudo-features learned with a contrastive loss on SAM masks. In contrast, our approach is simple to implement, applicable to any 2D representation and entirely parameter-free.

Leveraging 3D information to better segment in 2D. Most prior works focusing on semantic segmentation leverage 2D models specialized for this task. The early work of Yen-Chen et al. (2022) uplifts learned 2D image inpainters by optimizing view consistency over depth and appearance. Subsequent works have mostly relied on uplifting either features from SAM's encoder (Zhou et al., 2024), binary SAM masks (Cen et al., 2023c,b), or SAM masks automatically generated for all objects in the image (Ye et al., 2024; Ying et al., 2024; Kim et al., 2024). The latter approach is computationally expensive, as it requires querying SAM on a grid of points over the image. It also requires matching inconsistent mask predictions across views, with *e.g.* a temporal propagation model (Ye et al., 2024) or a hierarchical learning approach (Kim et al., 2024), which introduces additional computational overhead. In this work, we focus on single instance segmentation and show that our uplifted features are on par with the state of the art (Cen et al., 2023c,b; Ying et al., 2024). Standing out from prior work uplifting DINO features (Tschernezki et al., 2022; Kobayashi et al., 2022; Goel et al., 2023; Kerr et al., 2023; Liu et al., 2023b; Ye et al., 2023; Zuo et al., 2024), we show that DINOv2 features can be used on their own for semantic segmentation and rival SAM-based models through a simple graph diffusion process that leverages 3D geometry.

Learning 3D CLIP features for open-vocabulary object localization. For learning 3D CLIP features, prior works also leverage vision models such as DINO or SAM. DINO is used to regularize and refine CLIP features (Kerr et al., 2023; Liu et al., 2023b; Zuo et al., 2024; Shi et al., 2024), while SAM is employed for generating instance-level CLIP representations (Qin et al., 2024; Wu et al., 2024b; Jun-Seong et al., 2025). These approaches suffer from high computational costs, resorting to dimensionality reduction or efficient multi-resolution embedding representations, and usually run for a total of one to two hours for feature map generation and 3D feature optimization. In contrast, our approach bypasses the high computational cost of gradient-based optimization and, combined with graph diffusion, is an order of magnitude faster than these prior works.

5.3 Uplifting 2D visual representations into 3D

We now present a simple yet effective method for lifting 2D visual features into 3D using Gaussian Splatting, and discuss its relation with optimization-based techniques.

5.3.1 Background on Gaussian Splatting

Scene representation. The Gaussian Splatting method consists in modeling a 3D scene as a set of n Gaussians densities $\mathcal{N}_i$, each defined by a mean μ_i in $\mathbb{R}^3$, a covariance Σ_i in $\mathbb{R}^{3\times3}$, an opacity σ_i in $(0, 1)$, and a color function $c_i(d)$ that depends on the viewing direction d. d refers to the full camera pose, *i.e.* its extrinsics and intrinsics.

A 2D frame at a given view is an image $\hat{I}_d$ rendered by projecting the 3D Gaussians onto a 2D plane, parametrized by the viewing direction d. This projection accounts for the opacity of the Gaussians and the order in which rays associated with each pixel pass through the densities. More precisely, a pixel p for a view d is associated to an ordered set $\mathcal{S}_{d,p}$ of Gaussians and its value is obtained by summing their contributions:

$$\hat{I}_d(p) = \sum_{i\in\mathcal{S}_{d,p}} c_i(d)w_i(d,p). \tag{5.1}$$

The above weights result from α-blending, *i.e.* $w_i(d,p) = \alpha_i(d,p)\prod_{j\in\mathcal{S}_{d,p},j<i}(1-\alpha_j(d,p))$, where the Gaussian contributions $\alpha_i(d,p)$ are given by the product of the opacity σ_i and the Gaussian density $\mathcal{N}_i$ projected onto the 2D plane at pixel position p.

Scene optimization. Let $I_1,\dots,I_m$ be a set of 2D frames from a 3D scene and $d_1,\dots,d_m$ the corresponding viewing directions. Gaussian Splatting optimizes the parameters involved in the scene rendering function described in the previous section. This includes the means and covariances of the Gaussian densities, their opacities, and the color function parametrized by spherical harmonics. Denoting these parameters by θ, the following reconstruction loss is used

$$\min_\theta \frac{1}{m}\sum_{k=1}^{m}\mathcal{L}(I_k,\hat{I}_{d_k,\theta}), \tag{5.2}$$

where $\hat{I}_{d_k,\theta}$ is the frame of the scene rendered in the direction d_k as in Eq. (5.1), using the parameters θ, and $\mathcal{L}$ is a combination of ℓ_1 and SSIM loss functions (Kerbl et al., 2023).

5.3.2 Uplifting 2D feature maps into 3D

Given a 3D Gaussian Splatting scene representation, we propose a method to uplift any set of 2D features maps $\mathbf{F} = \{\mathbf{F_1},...,\mathbf{F_m}\}$ associated to m views into 3D. These features may be obtained from pretrained models (Radford et al., 2021; Oquab et al., 2024) or segmentation masks. The uplifted features $\mathbf{f}$ should allow for downstream 3D tasks, such as 3D segmentation, while being 'faithful' when reprojected back in 2D.

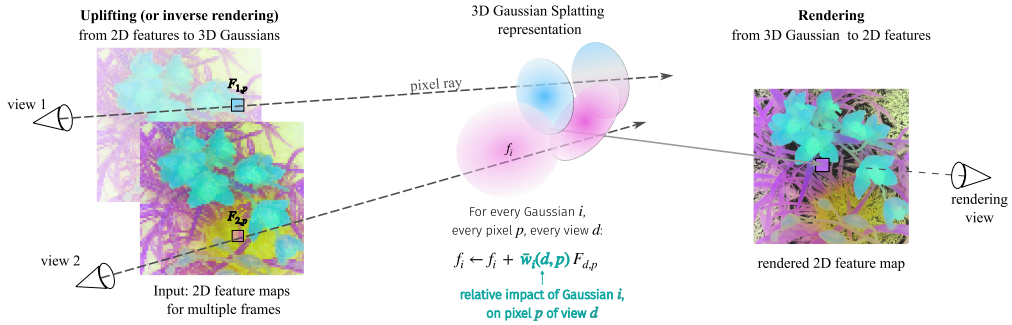


FIGURE 5.2: **Illustration of uplifting and rendering.** In the uplifting phase, features $\mathbf{f}$ are created for each 3D Gaussian by aggregating coarse 2D features $\mathbf{F}$ over all viewing directions. For rendering, the 3D features $\mathbf{f}$ are projected on any given viewing direction as in regular Gaussian Splatting. The rendering weight $\bar{w}_i(d,p)$ represents the relative influence of the Gaussian i on pixel p (Eq. (5.3)).

Uplifting with simple aggregation. We construct uplifted features for each 3D Gaussian of the Gaussian Splatting scene as a weighted average of 2D features from all frames. Each 2D feature $F_{d,p}$ from a frame at a given viewing direction d and pixel p contributes to the feature f_i by a factor proportional to the rendering weight $w_i(d,p)$, if the Gaussian i belongs to the ordered set $\mathcal{S}_{d,p}$ associated to the view/pixel pair (d,p). Denoting $\mathcal{S}_i = \{(d,p), i \in \mathcal{S}_{d,p}\}$ as the set of view/pixel pairs contributing to the feature f_i, the resulting features are defined as follows:

$$f_i = \sum_{(d,p)\in\mathcal{S}_i} \bar{w}_i(d,p) F_{d,p}, \quad \text{with} \quad \bar{w}_i(d,p) = \frac{w_i(d,p)}{\sum_{(d,p)\in\mathcal{S}_i} w_i(d,p)}. \tag{5.3}$$

We can interpret this equation as a normalized version of the transposed rendering operation over the m viewing directions. More precisely, the rendering of any view-independent collection of features $\mathbf{f} = (f_i)$ attached to the n Gaussians into the m training frames can be represented as a linear operator W acting on the collection $\mathbf{f}$ and returning a collection of 2D feature maps $\hat{\mathbf{F}} = (\hat{F}_{d,p})$, see Eq. (5.4) below. Here, non-zero entries of the matrix W consists of all rendering weights $w_i(d,p)$ when $(d,p) \in \mathcal{S}_i$ is placed at row (d,p) and column i, and $\hat{\mathbf{F}}$ is a 2D matrix containing all (flattened) 2D feature maps generated for all cameras poses, with $\hat{\mathbf{F}}_{d,p}$ the feature of pixel p from view at direction d. Similarly, the uplifting expression introduced in Eq. (5.3) can be expressed in terms of the transpose of W and a diagonal matrix D of size m representing the normalization factor and whose diagonal elements are obtained by summing over the rows of W as in Eq. (5.5) below:

Rendering to m frames

$$\hat{\mathbf{F}} = W\mathbf{f}, \tag{5.4}$$

Uplifting from m frames

$$\mathbf{f} = D^{-1}W^{\top}\mathbf{F}. \tag{5.5}$$

Note that W and D are never explicitly constructed. Instead, they are computed by calling the forward rendering function for Gaussian Splatting and replacing the color vectors by the feature vectors (see Fig. 5.2). All these operations are performed within the CUDA rendering process.

Connection with optimization-based inverse rendering. An alternative approach to uplifting 2D features $\mathbf{F}$ is to minimize a reconstruction objective $\mathcal{L}(\mathbf{f})$, where the goal is to find uplifted features $\mathbf{f}$ whose rendering closely matches the original 2D features $\mathbf{F}$ (Tschernezki et al., 2022; Kerr et al., 2023; Zuo et al., 2024). A natural choice is to minimize the mean squared error between the 2D features $\mathbf{F}$ and the rendered ones $\hat{\mathbf{F}}$ as defined by Eq. (5.4):

$$\min_{\mathbf{f}} \mathcal{L}(\mathbf{f}) := \frac{1}{2}\|\mathbf{F} - W\mathbf{f}\|^2. \tag{5.6}$$

Such an approach requires an optimization procedure which is costly compared to our proposed uplifting method. Nevertheless, it is possible to interpret the proposed uplifting scheme in Eq. (5.5) as a single pre-conditioned gradient descent step on the reconstruction objective, starting from a $\mathbf{0}$ feature, *i.e.*, $\mathbf{f} = -D^{-1}\nabla\mathcal{L}(\mathbf{0})$. In practice, we found that performing more iterations on the objective $\mathcal{L}(\mathbf{f})$ did not improve the quality of the features, thus suggesting that the computationally cheaper scheme in Eq. (5.5) is already an effective approach to uplifting.

Gaussian filtering. The normalization term $\beta_i = \sum_{d,p\in\mathcal{S}_i} w_i(d,p)$ serves as an estimator of the relative importance of each Gaussian in the scene. Therefore, it can be used as a criterion to prune the set of Gaussians for memory efficiency. In our experiments, we filter out half of the Gaussians based on β_i and observe no qualitative nor quantitative degradation of the results. This approach is inspired by prior work on efficient Gaussian Splatting representation such as proposed by Fan et al. (2023) who also prune Gaussians based on their contribution to each pixel in the training frames.

5.3.3 Enriching features by graph diffusion

DINOv2 features have shown remarkable performance on semantic segmentation with simple linear probing (Oquab et al., 2024), making them a good candidate to enrich features that lack such a property like CLIP (Wysoczańska et al., 2024; Zuo et al., 2024; Liu et al., 2023b). Inspired by spectral clustering techniques (Shi and Malik, 2000; Kondor and Lafferty, 2002; Belkin and Niyogi, 2001), we propose to *diffuse* features that have been uplifted to 3D. This process aims to align semantic features with the scene layout and object boundaries implied by DINOv2. In contrast to prior work, we

perform this landscaping with DINOv2 directly in the 3D scene, thereby taking 3D geometry into account as well.

Graph construction. We construct a graph whose nodes are the n 3D Gaussians and edges, represented by a matrix A of size $n \times n$, are based on 3D Euclidean geometry between the nodes and the similarity between their DINOv2 features. More precisely, we extract the k nearest neighbors $\mathcal{N}(i)$ for each node i, as measured by the Euclidean distance between the centers of the 3D Gaussians. Two nodes i and j in the graph are linked by an edge if $i \in \mathcal{N}(j)$ or $j \in \mathcal{N}(i)$, and the edge is assigned the following weight:

$$A_{ij} = S_f(f_i, f_j)\, P(f_i)^{\frac{1}{2}} P(f_j)^{\frac{1}{2}}, \tag{5.7}$$

with $S_f(f_i, f_j)$ a local similarity between features f_i and f_j, typically defined as a RBF kernel. For tasks requiring diffusion to be confined to a specific object instance, we prevent leakage into the background by introducing a node-wise unary regularization term $P(f_i)$ which quantifies the similarity between the node feature f_i and the features of the object of interest. Details on S_f and P are provided in the supplementary (Sec. C.1.3).

Diffusion on the graph. Given initial 3D features g_0 in $\mathbb{R}^n$, which we aim to improve by using information encoded in A (3D geometry and DINOv2 similarities), we perform T diffusion steps to construct a sequence of diffused features $(g_t)_{1 \leq t \leq T}$ defined as follows:

$$g_{t+1} = A\tilde{g}_t, \quad \tilde{g}_t = g_t / \|g_t\|_2, \tag{5.8}$$

This can be seen as performing a few steps of the power method, projecting g_0 into the dominant eigenspace of A. Depending on the task, g_0 may represent generic features or task-specific features such as 3D segmentation masks.

5.4 From 3D uplifting to downstream tasks

In this section, we describe our approach for extracting and uplifting features from SAM, DINOv2, and CLIP, as well as leveraging these features on two downstream tasks: multi-view segmentation and open-vocabulary object localization. As in Sec. 5.3, we are given a set of 2D frames $I_1, \ldots, I_m$, with viewing directions $d_1, \ldots, d_m$, and a 3D Gaussian Splatting representation of the scene.

5.4.1 Multi-view segmentation

We assume that a foreground mask of the object to be segmented is provided on the *reference frame* I_1. The foreground masks are either "scribbles" or a whole reference mask of the object, both of which define a set of foreground pixels $\mathcal{P}$. The task consists in generating a 2D segmentation mask on one or multiple *target labeled frames* based on the foreground from the *reference frame*. It is evaluated with the intersection over union (IoU).

Segmentation with SAM. SAM (Kirillov et al., 2023; Ravi et al., 2024) is a supervised pretrained model that, given an input image and point prompts, generates segmentation masks. Aggregating segmentation masks from SAM across multiple images in 3D improves cross-view consistency and hence single-view segmentation results. We proceed by generating 2D feature maps based on the SAM segmentation masks of each frame while only relying on the *foreground* for the reference frame I_1. The 2D SAM feature maps are obtained for each frame by averaging multiple mask predictions from several sets of point prompts. The point prompts are extracted by uplifting the *foreground mask* from the *reference frame* and reprojecting it onto the target frame, as described in the supplementary (Sec. C.1.1). The resulting feature maps are then uplifted using the aggregation scheme in Sec. 5.3.2. Our final prediction is obtained by rendering the uplifted feature maps into the target frame and thresholding. Results obtained with this strategy are reported in Table 5.1 for SAM (Kirillov et al., 2023) and SAM2 (Ravi et al., 2024).

Segmentation with DINOv2. We construct 2D feature maps at the patch level using DINOv2 with registers (Darcet et al., 2024) and uplift them into a high resolution and fine-grained 3D semantic representation which is then used for segmentation. The 2D feature maps are constructed using a combination of a sliding windows mechanism and dimensionality reduction of the original DINOv2 features as described in the supplementary (Sec. A.2). This approach enhances the granularity of spatial representations by aggregating patch-level representations to form pixel-level features. To favor the first principal components, known to focus on the foreground objects (Oquab et al., 2024), the features are re-weighted by the eigenvalues of the PCA decomposition. The 2D feature maps from the m training views are uplifted using Eq. (5.3) and the resulting 3D features are then re-projected into any viewing direction d using Eq. (5.4) to compute rendered 2D features ($\hat{F}_{d,p}$). To obtain segmentation masks, we define a predictor score $P(\hat{F}_{d,p})$ as the likelihood that a 2D pixel p belongs to the foreground, based on its feature $\hat{F}_{d,p}$. The score P is obtained by comparing the rendered features ($\hat{F}_{d,p}$) with foreground features $\mathcal{F}_{\text{ref}} := (\hat{F}_{d_1,p})_{p\in\mathcal{P}}$ corresponding to the foreground mask from the reference frame I_1, and the final segmentation mask is then

obtained by thresholding. More details are provided in the supplementary (Sec. C.1.2). Results with this approach correspond to an ablation presented in Table 5.2 (Uplifting, DINOv2) and show that it is already quite effective.

Improving DINOv2 segmentation with graph diffusion. Segmentation based on DINOv2 alone might struggle to distinguish distinct objects with similar features as it lacks 3D spatial information. We propose to incorporate this missing information by leveraging the graph diffusion process introduced in Sec. 4.4.3 and illustrated in Fig. 5.4, which achieves results competitive with SAM. We set the initial vector of weights $g_0 \in \mathbb{R}^n$ of the graph diffusion algorithm to be a coarse estimation of the contribution of each Gaussian to the final segmentation mask. This initial weight vector is computed by uplifting the 2D *foreground mask* (either scribbles or a reference mask) from the reference frame into 3D using Eq. (5.3), then normalizing and thresholding them (see supplementary Sec. C.1.3). The nodes for which g_0 has a positive value define a set of anchor nodes $\mathcal{M}$ that are more likely to contribute to the foreground. The regularization term P appearing in Eq. (5.7) is obtained by comparing the uplifted features with anchor features obtained using the *foreground mask* as described in the supplementary. For this task, we binarize A with a fixed threshold (set to 10^{-5}). After the T diffusion steps, we recover the nodes $\mathcal{S}$ in g_T with strictly positive values (*i.e.*, those reachable after T iterations). The final weight is defined as $h_i = P(f_i)$ if $i \in \mathcal{S}$ and 0 otherwise. Segmentation is then performed by projecting $\mathbf{h} = (h_i)$ into 2D and thresholding. Our results with this approach are reported as *DINOv2* in Table 5.1 and *Uplifting + Dif.* in Table 5.2.

5.4.2 Open-vocabulary object localization

Following Kerr et al. (2023), we tackle the task of open-vocabulary object localization by uplifting CLIP features (Ilharco et al., 2021). CLIP effectively aligns images and text in a shared representation space. As a measure of alignment, we use the relevancy score introduced by LERF (Kerr et al., 2023), which measures the similarity between a CLIP visual feature and a text query.

Construction of CLIP feature maps. We follow common practice (Kerr et al., 2023; Zuo et al., 2024) and construct multi-resolution CLIP 2D feature maps by querying CLIP on a grid of overlapping patches at different scales and aggregating the resulting representations. As Zuo et al. (2024), rather than keeping the different representations separate, we aggregate them with a simple average pooling. These multi-resolution CLIP features are uplifted into 3D using Eq. (5.3).

Relevancy scores. After uplifting CLIP features, we compute relevancy scores for each Gaussian's feature to text queries embedded by CLIP. These relevancy scores can then be projected into 2D and used for both localization and segmentation. For localization, we choose the pixel with the highest relevancy score. For segmentation, we render the relevancy scores and either (i) directly threshold the rendered scores, or (ii) predict a SAM mask by selecting point prompts among pixels with the highest score. Specifics on the computation of relevancy scores and segmentation masks are provided in the supplementary (Sec. C.1.4).

Refining relevancy scores with DINOv2 graph diffusion. We refine 3D relevancy scores with the diffusion process described in Sec. 4.4.3. To this end, DINOv2 features are also uplifted, and the similarity matrix is built as in Eq. (5.7), with the unary term P constructed using a logistic regression over thresholded relevancies, see details in the supplementary (Sec. C.1.4). The diffusion process propagates CLIP relevancies to Gaussians with similar DINOv2 features. The resulting 3D relevancy scores span the object of interest without covering other objects with similar features and show a strong decay at the object's borders, as defined by DINOv2's feature landscape, resulting in improved segmentation results. This approach is evaluated and compared to direct segmentation without diffusion in Sec. 5.5.4.

5.5 Experiments

5.5.1 Experiment details

3D scene training and pruning. All scenes are trained using the original Gaussian Splatting implementation (Kerbl et al., 2023) with default hyperparameters. For memory efficiency, half of the Gaussians are filtered out based on their importance, as described in Sec. 5.3.2.

2D vision models. Our experiments are conducted using DINOv2's ViT-g with registers (Darcet et al., 2024), SAM (Kirillov et al., 2023), SAM 2 (Ravi et al., 2024) and the OpenCLIP ViT-B/16 model (Ilharco et al., 2021).

Segmentation. We consider two segmentation tasks: (i) Neural Volumetric Object Selection (NVOS, Ren et al. 2022), which is derived from the LLFF dataset (Mildenhall et al., 2019), and (ii) SPIn-NeRF, which contains a subsets of scenes from NeRF-related datasets (Knapitsch et al., 2017; Mildenhall et al., 2019, 2021; Yen-Chen et al., 2022; Fridovich-Keil et al., 2022). NVOS consists of forward-facing sequences where the task is to predict the segmentation mask of a labeled frame based on reference scribbles

from another frame. SPIn-NeRF contains both forward-facing and 360-degree scenes, in which all frames are labeled with segmentation masks, and the standard evaluation protocol uses the segmentation mask from the first frame as reference to label the subsequent frames.

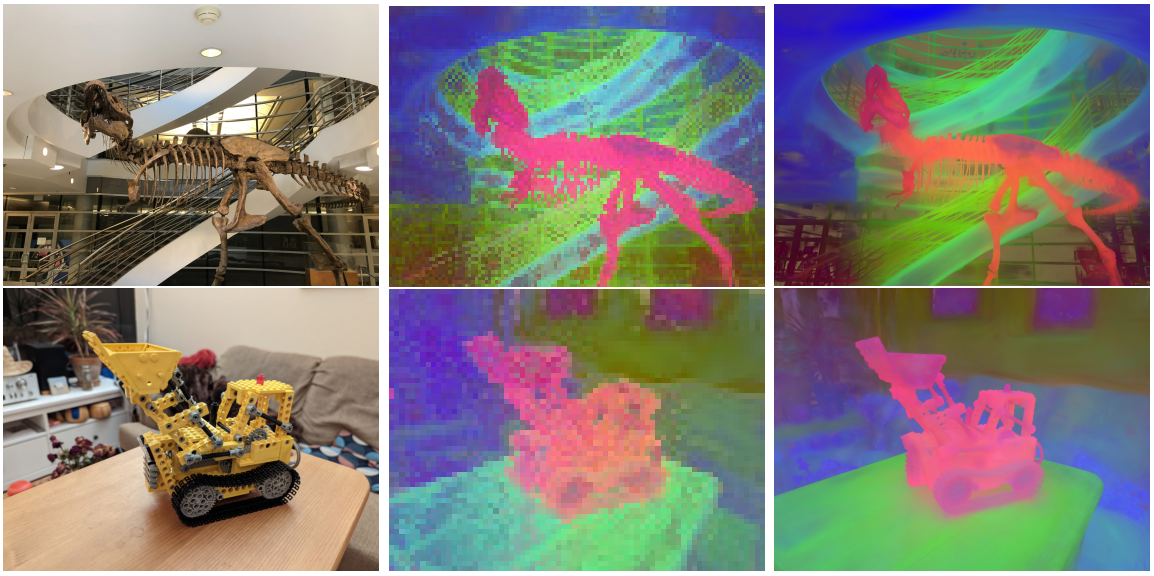
Open-vocabulary object localization. We evaluate on the LERF dataset (Kerr et al., 2023), which includes localization and segmentation tasks on complex in-the-wild scenes. We report our results on the extended evaluation task introduced by LangSplat (Qin et al., 2024) containing additional challenging localization samples, and also follow their evaluation protocol.

Evaluation. Results are averaged over 3 independent runs. The key parameters across tasks include thresholding for segmentation and those defining S_f and P in graph diffusion. These parameters are either fixed for all scenes in a task or automatically selected, as detailed in the supplementary (Sec. C.1).

5.5.2 Qualitative results

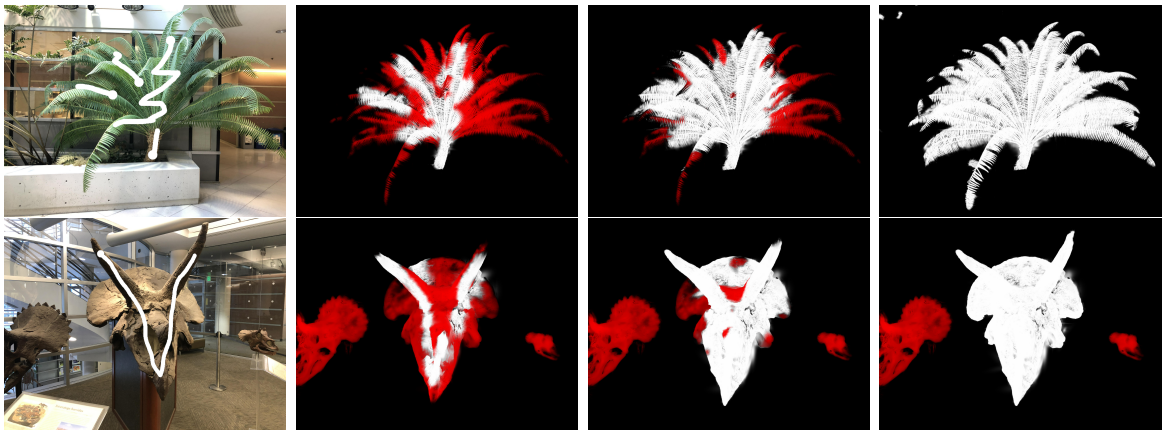
DINOv2 feature uplifting. First, we illustrate the effectiveness of our simple uplifting approach. Fig. 5.3 shows the first three PCA components (one channel per component) over DINOv2's patch embeddings. The coarse patch-level representations from every view (middle) are aggregated using Eq. (5.5) to form a highly detailed 3D semantic representation, and reprojected into 2D (right) using Eq. (5.4). The aggregation is very fast, being directly implemented in the Gaussian Splatting CUDA-based rendering process, and takes about 2ms per view and feature dimension. The first principal component (encoded in the red channel) mostly captures the foreground object, and the subsequent ones allow refining the foreground representations and delivering a detailed background. In the supplementary, we provide additional comparative visualizations of our learned 3D features and of 3D segmentation for object removal.

Graph diffusion. Fig. 5.4 illustrates the diffusion process. The graph nodes are initialized with the reference scribbles, and the diffusion spreads through the object of interest. As illustrated with the case of Horns, diffusion filters out unwanted objects that are similar to the object of interest (here, the two skulls on the side). In the Fern scene, diffusion progressively spreads through the branches to their extremities, with the regularization (red background) constraining it to the trunk and preventing leakage, even after a large number of iterations. Fig. C in the supplementary also illustrates this for the Flower and Trex scenes: diffusion rapidly spreads, achieving near-full coverage after only 5 steps before reaching all the much smaller Gaussians on the border, allowing for a refined segmentation.



(A) RGB image (B) Single-view PCA (C) Multi-view PCA

FIGURE 5.3: **PCA visualizations.** The DINOv2 patch-level representations (middle) predicted from the RGB images (left) are aggregated into highly detailed 3D representations (right) using Eq. (5.3).



(A) RGB image (B) 3 steps (C) 7 steps (D) 100 steps

FIGURE 5.4: **Diffusion process.** 2D projection of the weight vector g_t (white) and unary regularization term (red) at diffusion steps t. The diffusion process filters out unwanted objects that have similar features to the object of interest (such as the two smaller skulls on *horns*, bottom-row), but are disconnected in space. The regularization term (red background) prevents leakage from the object to the rest of the scene (such as through the *fern*'s trunk, top-row).

Open-vocabulary object removal. Fig. 5.1 and 5.5 provide qualitative results for the object removal task, performed by discarding all Gaussians corresponding to a binary 3D segmentation mask. This mask is obtained by computing relevancy between 3D CLIP features and the scene-specific textual prompt "bonsai in a ceramic pot" or "teddy bear", followed by DINOv2-guided graph diffusion.

FIGURE 5.5: **Open-vocabulary object removal.** Removing the teddy bear from the CO3D dataset (Reizenstein et al., 2021), using DINOv2-guided graph diffusion based on 3D CLIP relevancy scores.

5.5.3 Multi-view segmentation results

In this section, we quantitatively evaluate the multi-view segmentation task as defined in NVOS (Ren et al., 2022) and SPin-NeRF (Mirzaei et al., 2023). We evaluate segmentation based on SAM and SAM2 mask uplifting and on DINOv2 feature uplifting combined with graph diffusion (see Sec. 5.4.1). We compare our segmentation results (IoU) to the current state of the art: SA3D (Cen et al., 2023c), SA3D-GS (Cen et al., 2023b), SAGA (Cen et al., 2023a), OmniSeg3D (Ying et al., 2024). All these methods are specifically designed for uplifting the 2D segmentation masks produced by SAM into 3D using gradient-based optimization of a projection loss. We also report results from NVOS (Ren et al., 2022) and MVSeg (Yen-Chen et al., 2022), who respectively introduced the NVOS and SPIn-NeRF datasets.

Results. Table 5.1 reports the average IoU across all scenes for NVOS and SPIn-NeRF. Per-scene results can be found in supplementary Tables C.2 and C.3. Our results are comparable to the state of the art, while not relying on gradient-based optimization. Surprisingly, our segmentation with DINOv2 using graph diffusion also gives results on par with models leveraging SAM masks. Compared to SAM, DINOv2 better captures complex objects, but sometimes also captures some background noise. This can be seen in supplementary Fig. C.2 with the example of the T-Rex: while SAM misses out the end of the tail as well as the end of the ribs, DINOv2 captures the whole Trex, but also captures part of the stairs behind. Our lower segmentation results compared to OmniSeg's can partly be attributed to the poor Gaussian Splatting reconstruction of highly specular scenes, such as Fork. As also noted by Cen et al. (2023a), the reconstruction includes semi-transparent Gaussians floating over the object, attempting to represent reflections or surface effects, which are challenging to capture using standard rasterization techniques (Jiang et al., 2024).

	NVOS	MVSeg	SA3D-TRF	SA3D-GS	SAGA	OmniSeg3D	LUDVIG (Ours)		
3D representation:			TensoRF	GS	GS	NeRF	GS		
Uplifting:			SAM	SAM	SAM	SAM	DINOv2	SAM	SAM2
NVOS	70.1	-	90.3	92.2	92.6	91.7	92.4	91.3	91.3
SPIn-NeRF	-	90.9	93.7	93.2	93.4	94.3	93.8	93.8	93.8

TABLE 5.1: Segmentation (IoU) on NVOS (Ren et al., 2022) and SPIn-NeRF (Mirzaei et al., 2023), compared against prior works.

Geometry only	Single view		Uplifting		Uplifting + Graph diffusion
Reference mask	DINOv2	SAM2	DINOv2	SAM2	DINOv2
73.1	88.5	88.6	91.6	93.8	93.8

TABLE 5.2: **Ablation for segmentation on SPIn-NeRF (Mirzaei et al., 2023).** We compare purely geometrical reference mask uplifting and reprojection, single-view prediction, feature or mask uplifting, and graph diffusion leveraging DINOv2 or SAM2.

	ramen	figurines	teatime	waldo	**overall**
LERF (Kerr et al., 2023)	62.0	75.0	84.8	72.7	73.6
LangSplat (Qin et al., 2024)	73.2	80.4	88.1	**95.5**	84.3
LUDVIG (ours)	**78.9**	**80.4**	**94.9**	90.9	**86.3**

TABLE 5.3: **LERF object localization.** Comparison with the state of the art on the dataset introduced by LangSplat (Qin et al., 2024).

Ablation study. We compare our segmentation protocol using DINOv2 and SAM2 to multiple simpler variants. More precisely, we evaluate (i) a purely geometrical variant that reprojects the reference mask on the other views, without using SAM2 or DINOv2, (ii) single-view segmentation in 2D based on SAM2 or DINOv2 2D predictions, (iii) uplifting DINOv2 features or SAM2 masks into 3D then rendering them for segmentation, and (iv) segmenting using graph diffusion over DINOv2 3D feature similarities. Results are reported in Table 5.2, and per-scene IoU as well as a detailed analysis can be found in supplementary Table C.4 and Sec. C.2.2. We observe that the purely geometrical approach works well on the forward-facing scenes and fails on 360-degree scenes. The single-view variant performs reasonably well on average, but the low resolution of patch-level representations (illustrated in Fig. 5.3) lead to a coarser segmentation. 3D uplifting considerably boosts results compared to single-view approaches, and introducing 3D spatial information through 3D graph diffusion further enhances results on the more challenging 360-degree scenes.

5.5.4 Open-vocabulary object localization

Tables 5.3 and 5.4 present results on the open-vocabulary LERF localization and segmentation tasks (Kerr et al., 2023), evaluated on the extended and more challenging

	ramen	figurines	teatime	waldo	**overall**	time (min)
Initial evaluation protocol from Kerr et al. (2023); Qin et al. (2024): 2D object segmentation						
LERF (Kerr et al., 2023)	28.2	38.6	45.0	37.9	37.4	45
LangSplat (Qin et al., 2024)	51.2	44.7	65.1	44.5	51.4	105
LUDVIG (ours)	**58.1**	**63.3**	**77.1**	**58.5**	**64.3**	10
New evaluation protocol introduced in OpenGaussian (Wu et al., 2024b): 3D object selection						
OpenGaussian (Wu et al., 2024b)	23.9	59.3	54.4	34.6	43.1	50
Dr. Splat (Jun-Seong et al., 2025)	24.7	53.4	57.2	39.1	43.6	-
LUDVIG (ours)	**42.3**	**58.0**	**58.6**	**42.8**	**50.4**	10

TABLE 5.4: **LERF object segmentation.** We evaluate on the extended LERF dataset introduced by LangSplat (Qin et al., 2024) with two different evaluation protocols: i) segmentation based on the 2D reprojected features, ii) 3D segmentation (or *object selection* and reprojection of the *binary 3D masks*, proposed by Kerr et al. (2023); Qin et al. (2024) and Wu et al. (2024b) respectively.

SAM	Graph diffusion	ramen	figurines	teatime	waldo	overall
✗	✗	27.8	37.8	38.2	30.4	33.6
✗	✓	42.3	58.0	58.6	42.9	50.4
✓	✗	52.2	51.8	68.9	56.4	57.3
✓	✓	**58.1**	**63.3**	**77.1**	**58.5**	**64.3**

TABLE 5.5: **Ablation for LERF segmentation.** Results (IoU) with/without 3D graph diffusion and/or 2D SAM segmentation, evaluated on the dataset introduced by LangSplat (Qin et al., 2024).

dataset introduced by LangSplat (Qin et al., 2024). Table 5.4 also reports average running times, which include feature map generation and 3D feature training whenever relevant. Overall, LUDVIG outperforms prior works while being significantly faster than all methods (5 to 10 times faster). The supplementary provides more analysis of the run times and additional qualitative results. Below we detail and discuss our results for Tables 5.4 and 5.5.

Open-vocabulary object segmentation. We consider two evaluation protocols presented in Table 5.4: (i) *standard 2D segmentation*, as introduced by Kerr et al. (2023) (top part of the table), and (ii) *3D object selection*, which consists of binarizing the 3D masks before rendering (bottom part). The latter protocol was introduced by Jun-Seong et al. (2025) as a way to better assess the quality of 3D semantic features, such as 3D CLIP features, compared to prior existing protocols which focus on the performance on 2D downstream tasks.

For both approaches, we refine raw 3D CLIP relevancy scores using graph diffusion and use them for segmentation as follows. For (i) we perform 2D segmentation from rendered features using SAM, as described in Sec. 5.4.2, which we compare to 2D segmentation by LERF (Kerr et al., 2023) and LangSplat (Qin et al., 2024) (top of table).

For (ii) we directly segment in 3D by binarizing the diffused features before rendering them, then computing 2D segmentation masks from these rendered features. We use automatic thresholding for binarization before and after rendering. Our 3D segmentation results based on graph diffusion surpass those obtained by OpenGaussian (Wu et al., 2024b) and Dr. Splat (Jun-Seong et al., 2025) (bottom of table).

Ablation study. Table 5.5 presents an ablation study on the effect of graph diffusion and SAM-based segmentation following the first evaluation protocol (top part) in Table 5.4.

5.6 Concluding remarks and limitations

In this work, we introduce a simple yet effective aggregation mechanism for transferring 2D visual representations into 3D, bypassing traditional optimization-based approach. The aggregation builds upon already trained Gaussian Splatting representations and is implemented within the CUDA rendering process, making 2D-to-3D uplifting as fast as 3D-to-2D rendering. Note, however, that the quality of 3D features is bound by that of the 3D scene reconstruction which is notoriously challenging for, *e.g.*, highly specular (Jiang et al., 2024; Yang et al., 2024) or blurry (Zhao et al., 2024; Lee et al., 2024) scenes. In such scenarios, learning 3D features *along with* 3D Gaussian Splatting reconstruction may improve scene geometry, opening promising perspectives for future work.

After feature uplifting, our graph diffusion process leverages spatial information together with the rich DINOv2 representations to transform coarse 2D or 3D segmentation signals, such as scribbles or CLIP relevancy scores, into accurate 3D segmentation masks. Our CLIP relevancy refinement builds upon prior works using DINO features as a regularization (Kerr et al., 2023; Zuo et al., 2024), while alleviating the computational overhead associated with joint gradient-based optimization of CLIP and DINO features. Our approach significantly outperforms SAM-based prior works in open-vocabulary 3D object segmentation while relying only on generic CLIP and DINOv2 features.

Chapter 6

Conclusion

A central question explored throughout this thesis is the following: Given a specific task, how can one make the most effective use of the available data, computational resources, and general-purpose pretrained models? This question underlies all three axes explored in this manuscript: (i) automatically selecting data augmentation policies, (ii) distilling task-specific knowledge from large models into smaller ones, and (iii) integrating semantic understanding into 3D scene representations. Each of these contributions addresses a different facet of a broader challenge: enabling visual representation learning under practical, often constrained, real-world conditions.

Efficient training and deployment

Improving the quality of learned representations is a central goal in computer vision. Equally important, however, is the challenge of operating under real-world computational constraints. This section examines how recent advances—and our own contributions—promote efficiency across training, adaptation, and deployment.

Lightweight model adaptation. As large general-purpose vision models become increasingly prevalent, a growing body of work explores how to adapt them efficiently to new tasks. Rather than updating all parameters, recent methods introduce small task-specific modules while keeping the backbone frozen. Adapter layers (Houlsby et al., 2019; Chen et al., 2022b), low-rank reparameterizations (Hu et al., 2022), and scaling-based approaches like IA^3 (Liu et al., 2022a) all achieve strong results with minimal overhead. These methods have had a broad impact, enabling scalable deployment across a wide range of tasks and resource settings. More generally, this reflects a growing emphasis on scaling down the cost of adaptation, complementing the continued trend toward increasing model capacity.

Lightweight model deployment. Once a model has been trained or adapted, its usability in real-world settings often hinges on inference-time efficiency. Applications may require fast, low-memory, or on-device inference, where large models become impractical. To address this, techniques such as pruning (Zhu and Gupta, 2018), quantization (Jacob et al., 2018), and knowledge distillation are widely used to compress

large models into smaller, general-purpose variants (Wu et al., 2022; Oquab et al., 2024). Knowledge distillation has emerged as perhaps the most prominent approach for model compression in vision, with pretrained models such as DINOv2's ViT-S offering state-of-the-art performance among compact models.

However, general-purpose distillation typically focuses on preserving the teacher's overall representational space, often at the cost of performance drop on specific downstream tasks. This motivates task-specific distillation, where knowledge is selectively transferred to optimize for the downstream objective. Chapter 4 explores best practices for task-specific knowledge distillation. We show that probing a pretrained model—rather than finetuning it—can yield more effective teachers in certain settings, offering a lighter and more robust alternative. We also find that task-specific distillation complements task-agnostic approaches: while generic distillation transfers broad representational power, aligning student and teacher predictions for the downstream objective provides an additional performance boost. Together, these results point to a practical, efficient strategy for deploying compact, task-specialized models that benefit from large pretrained teachers without relying on them at inference.

Label-efficient model deployment. A major obstacle to adapting models in practice is the limited availability of labeled data. In this manuscript, we explore complementary lines of research for addressing this challenge. In Chapter 3, we propose a method for automatically learning data augmentation policies via a bilevel optimization framework. Our approach stabilizes and regularizes the optimization process, enabling the discovery of optimal augmentation strategies without relying on hard-coded transformations. In Chapter 4, we investigate synthetic images as a form of data augmentation for task-specific knowledge distillation. In particular, we show that diffusion-generated images—without the need for class-level prompts—can serve as a rich and flexible augmentation source, expanding the set of samples used to align student and teacher predictions.

These contributions align with a growing body of work aimed at improving generalization with limited labeled data by introducing additional forms of supervision. With knowledge distillation, the teacher's predictions serve as soft labels providing an additional source of supervision; this paradigm has been extensively adopted in related research areas such as continual learning (Fini et al., 2022; Wang et al., 2024b), where it helps mitigate forgetting by preserving knowledge across tasks. Similarly, synthetic image generation has shown strong potential in low-label regimes, offering a way to expand the training distribution without additional annotations (Trabucco et al., 2023).

While these data-driven approaches focus on enriching the training data or supervisory

signal, a complementary direction involves embedding stronger inductive biases into the model itself. Equivariance is a particularly impactful bias in domains such as molecular modeling, physics, and 3D geometry, where data is scarce and symmetries are intrinsic (Bronstein et al., 2021; Satorras et al., 2021). By aligning model behavior with known transformation groups, equivariant architectures can generalize more effectively with limited supervision. Though less common in general-purpose vision systems, such structure-aware design offers a promising avenue for label-efficient learning—especially in domains where supervision is limited but structure is rich.

Hardware-aware developments. Recent progress in training and inference efficiency has been largely driven by advances that optimize memory access and exploit parallelism. These improvements are central to reducing both computational cost and memory usage during training and inference, and have shaped many of the most impactful developments in the field. The attention mechanism (Vaswani et al., 2017), for instance, is naturally parallelizable across multiple heads, making it well suited to large-batch training and scalable computation. Recent improvements such as FlashAttention (Dao et al., 2022) heavily reduce memory access and computational overhead by restructuring attention into fused GPU kernels. These improvements yield substantial speedups and memory savings, especially for long sequences and deep models. Low-level optimizations—such as operator fusion in custom CUDA attention kernels and blockwise execution strategies, as implemented in libraries like xFormers (Lefaudeux et al., 2022)—follow a similar principle: aligning algorithmic design with hardware capabilities to maximize performance and reduce memory overhead. This exemplifies an ongoing shift toward reducing memory access as a primary bottleneck in training efficiency (Dao et al., 2022).

A similar shift toward hardware-conscious design is visible in 3D scene representations. Traditional neural rendering techniques like NeRF (Mildenhall et al., 2021) rely on volumetric ray marching, which is accurate but slow and hard to scale. Gaussian Splatting (Kerbl et al., 2023) responds to this limitation by using anisotropic 3D Gaussians as primitives, rendered through rasterization rather than ray integration. This allows for real-time synthesis while preserving high-fidelity geometry and appearance—demonstrating that rethinking the representation itself, with hardware constraints in mind, can unlock orders-of-magnitude speedups.

Learning semantics beyond 2D images

While general-purpose models have flourished in 2D vision and language, their counterparts in other modalities—such as 3D vision—remain comparatively underdeveloped. This stems not only from the lack of large-scale, diverse 3D datasets—which played a crucial role in the success of models like DINOv2 or CLIP—but also from the difficulty of learning general-purpose representations for structured data like 3D scenes. In practice, most pipelines still rely on training a 3D representation independently for each scene, typically supervised via reprojection losses from 2D images.

This has motivated a growing body of research focused on transferring features from pretrained 2D models into 3D. Works such as Kerr et al. (2023); Cen et al. (2023c); Chen et al. (2024); Kobayashi et al. (2022); Fan et al. (2023); Ye et al. (2024) uplift 2D features extracted from models like DINOv2 (Oquab et al., 2024), CLIP (Radford et al., 2021), or SAM (Kirillov et al., 2023) into 3D scene representations to enable tasks such as 3D object retrieval, segmentation, and editing. These approaches usually learn 3D features by optimizing a reprojection loss through gradient descent. This process is (i) computationally expensive—especially for high-dimensional features—and (ii) does not guarantee spatial consistency in 3D, as supervision is applied only in image space. These limitations are well documented in recent works such as OpenGaussian (Wu et al., 2024b) and DrSplat (Jun-Seong et al., 2025), which report degraded 3D localization performance and weak geometric coherence in the resulting features.

In Chapter 5, we propose a simple, learning-free method for equipping already trained Gaussian Splatting (GS) representations (Kerbl et al., 2023) with semantics. We project 2D features from multiple views onto each 3D Gaussian using an alpha-weighted aggregation strategy, producing semantic descriptors without relying on gradient-based optimization. Despite its simplicity, this approach produces 3D features that support direct segmentation in 3D space, indicating that the resulting representation captures meaningful semantic structure.

To leverage this structure, we introduce a graph diffusion mechanism that propagates semantic information across Gaussians using 3D geometry and feature similarity. This enables object-centric reasoning in the 3D Gaussian space, transforming coarse semantic signals—such as user scribbles or CLIP relevancy scores—into detailed segmentation masks. Our approach eliminates the need for pipelines that combine 2D segmentation with view-consistent mask uplifting.

However, the quality of our 3D features fundamentally depends on the underlying GS representation. Since no learning is applied during semantic uplift, any inaccuracies in geometry or appearance directly affect the projected features. Training GS scenes

remains challenging under difficult conditions such as motion blur, limited view coverage, high-frequency textures, or inconsistent lighting. The semantic expressiveness of the uplifted features is thus bounded by the fidelity of the reconstructed scene.

Beyond scene-level reasoning, a major open challenge is generalization. Unlike in 2D vision—where large pretrained models like DINOv2 or CLIP can be used across tasks and datasets without retraining—dominant pipelines in 3D vision, such as NeRF (Mildenhall et al., 2021) and Gaussian Splatting (Kerbl et al., 2023), still require training a separate representation for every new scene. This severely limits scalability, especially in applications that demand rapid adaptation or operate in dynamic environments.

Recent works have begun to address this limitation, not by building fully general-purpose 3D representations, but by proposing architectures that sidestep the need for dense optimization altogether. DUSt3R (Wang et al., 2024c), for instance, shows that accurate local geometry and camera pose estimation can be recovered directly from image pairs—even in the absence of known intrinsics—by regressing per-pixel depth in a relaxed projective setup. Rather than representing scenes with learned volumetric fields, it produces explicit pointmaps and framewise depth, and relies on robust geometric priors rather than scene-level learning. Its impact has been substantial: not only does it unify monocular and stereo settings under a single formulation, but it has also sparked a wave of rethinking on how 3D geometry can be recovered from unposed images. MASt3R (Leroy et al., 2024) builds on this idea by grounding matching in 3D: instead of computing correspondences purely in 2D, it leverages the depth-aware structure of DUSt3R to define dense features that are consistent across viewpoints. Its strong performance on tasks such as map-free localization demonstrates the benefit of 3D-first representations, especially in challenging, real-world settings with wide viewpoint baselines. While these methods do not yet produce reusable 3D scene representations in the way a model like DINOv2 does for images, they mark an important shift: they show that multi-view geometry can be recovered at inference time, without scene-specific training, and that structural priors can replace some of the learning burden.

Together, these approaches point toward a new generation of 3D vision systems that may not require end-to-end scene-specific training but instead fuse rapid geometric reasoning with semantic knowledge from large 2D models. This shift suggests that the future of 3D vision may lie in hybrid pipelines that combine efficient, structure-aware inference with the representational richness of pretrained 2D systems to enable scalable and adaptable scene understanding.

Bibliography

Soroush Abbasi Koohpayegani, Ajinkya Tejankar, and Hamed Pirsiavash. Compress: Self-supervised learning by compressing representations. In Advances in Neural Information Processing Systems (NeurIPS), 2020. 23, 47, 60

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. In Advances in Neural Information Processing Systems (NeurIPS), 2022. 18

Michael Arbel and Julien Mairal. Non-convex bilevel games with critical point selection maps. preprint arXiv:2207.04888, 2022. 4, 27

Shekoofeh Azizi, Simon Kornblith, Chitwan Saharia, Mohammad Norouzi, and David J Fleet. Synthetic data from diffusion models improves imagenet classification. Transactions on Machine Learning Research (TMLR), 2023. 8, 16, 44, 48

Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In Advances in Neural Information Processing Systems (NIPS), 2014. 47

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. arXiv preprint arXiv:1607.06450, 2016. 2

Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEit: BERT pre-training of image transformers. In Proceedings of the International Conference on Learning Representations (ICLR), 2022. 3, 17

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In Proceedings of the European Conference on Computer Vision (ECCV), 2006. 1

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. Advances in neural information processing systems (NIPS), 2001. 10, 65, 70

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In Advances in Neural Information Processing Systems (NeurIPS), 2006. 2

Lucas Beyer, Xiaohua Zhai, Amélie Royer, Larisa Markeeva, Rohan Anil, and Alexander Kolesnikov. Knowledge distillation: A good teacher is patient and consistent. In

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022. 8, 22, 23, 44, 45, 47, 48, 54

Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. arXiv preprint arXiv:2104.13478, 2021. 83

Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023. 48

Yohann Cabon, Lucas Stoffl, Leonid Antsfeld, Gabriela Csurka, Boris Chidlovskii, Jerome Revaud, and Vincent Leroy. Must3r: Multi-view network for stereo 3d reconstruction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2025. 21

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In Proceedings of the European Conference on Computer Vision (ECCV), 2018. 2, 17

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In Advances in Neural Information Processing Systems (NeurIPS), 2020. 2, 17, 18

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In Proceedings of the International Conference on Computer Vision (ICCV), 2021. 2, 9, 17, 18, 22, 64, 66

Jiazhong Cen, Jiemin Fang, Chen Yang, Lingxi Xie, Xiaopeng Zhang, Wei Shen, and Qi Tian. Segment any 3d gaussians. arXiv preprint arXiv:2312.00860, 2023a. 77

Jiazhong Cen, Jiemin Fang, Zanwei Zhou, Chen Yang, Lingxi Xie, Xiaopeng Zhang, Wei Shen, and Qi Tian. Segment anything in 3d with radiance fields. arXiv preprint arXiv:2304.12308, 2023b. 22, 23, 66, 67, 77, 142

Jiazhong Cen, Zanwei Zhou, Jiemin Fang, Wei Shen, Lingxi Xie, Dongsheng Jiang, Xiaopeng Zhang, Qi Tian, et al. Segment anything in 3d with nerfs. In Advances in Neural Information Processing Systems (NeurIPS), 2023c. 9, 10, 64, 66, 67, 77, 84

Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In Proceedings of the European Conference on Computer Vision (ECCV), 2022a. 21

Defang Chen, Jian-Ping Mei, Yuan Zhang, Can Wang, Zhe Wang, Yan Feng, and Chun Chen. Cross-layer distillation with semantic calibration. In Proceedings of the AAAI Conference on Artificial Intelligence, 2021a. 47

Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587, 2017. 53

Pengguang Chen, Shu Liu, Hengshuang Zhao, and Jiaya Jia. Distilling knowledge via knowledge review. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021b. 47

Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. In Advances in Neural Information Processing Systems (NeurIPS), 2022b. 7, 81

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Proceedings of the International Conference on Machine Learning (ICML), 2020a. 2, 16

Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. Uniter: Universal image-text representation learning. In Proceedings of the European Conference on Computer Vision (ECCV), 2020b. 18

Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024. 9, 23, 64, 66, 84, 148, 149

Jang Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation. In Proceedings of the International Conference on Computer Vision (ICCV), 2019. 60, 61

Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In Proceedings of the European Conference on Computer Vision (ECCV), 2016. 19

Jaeyoung Chung, Jeongtaek Oh, and Kyoung Mu Lee. Depth-regularized optimization for 3d gaussian splatting in few-shot images. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024. 145

M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2014. 52, 125, 133, 134

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2016. 52, 125

Corinna Cortes. Support-vector networks. Machine Learning, 1995. 1, 14

Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In Workshop on statistical learning in computer vision, ECCV, 2004. 1, 13, 14

Ekin D. Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation strategies from data. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019. 4, 15, 28, 33, 37, 114

Ekin Dogus Cubuk, Barret Zoph, Jon Shlens, and Quoc Le. RandAugment: Practical automated data augmentation with a reduced search space. In Advances in Neural Information Processing Systems (NeurIPS), 2020. 5, 29, 37, 114

Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In Proceedings of the ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques, 1996. 19

Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2017. 9

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2005. 1, 13

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In Advances in Neural Information Processing Systems (NeurIPS), 2022. 83

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In Proceedings of the International Conference on Learning Representations (ICLR), 2024. 10, 65, 72, 74, 136

Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In Proceedings of the ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques, 1996. 2, 21

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2009. 16, 26

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers), 2019. 17, 18

Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552, 2017. 27

Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929, 2020. 2, 14, 16

Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In Proceedings of the European Conference on Computer Vision (ECCV), 2020. 22

Bardienus Duisterhof, Lojze Zust, Philippe Weinzaepfel, Vincent Leroy, Yohann Cabon, and Jerome Revaud. Mast3r-sfm: a fully-integrated solution for unconstrained structure-from-motion. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2025. 20

Lisa Dunlap, Alyssa Umino, Han Zhang, Jiezhi Yang, Joseph E Gonzalez, and Trevor Darrell. Diversify your vision datasets with automatic diffusion-based augmentation. In Advances in Neural Information Processing Systems (NeurIPS), 2024. 48

Quentin Duval, Ishan Misra, and Nicolas Ballas. A simple recipe for competitive low-compute self supervised vision models. arXiv preprint arXiv:2301.09451, 2023. 23, 47

Dave Epstein, Allan Jabri, Ben Poole, Alexei A Efros, and Aleksander Holynski. Diffusion self-guidance for controllable image generation. In Advances in Neural Information Processing Systems (NeurIPS), 2023. 48

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. International Journal of Computer Vision (IJCV), 88(2):303–338, 2010. 52, 125

Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. arXiv preprint arXiv:2311.17245, 2023. 9, 64, 70, 84

Haoyang Fang, Boran Han, Shuai Zhang, Su Zhou, Cuixiong Hu, and Wen-Ming Ye. Data augmentation for object detection via controllable diffusion models. In Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV), 2024. 48

Yuxin Fang, Quan Sun, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. EVA-02: A visual representation for neon genesis. arXiv preprint arXiv:2303.11331, 2023. xiv, 6, 44, 45, 60, 125, 131, 132

Zhiyuan Fang, Jianfeng Wang, Lijuan Wang, Lei Zhang, Yezhou Yang, and Zicheng Liu. SEED: Self-supervised distillation for visual representation. In Proceedings of the International Conference on Learning Representations (ICLR), 2021. 22, 23, 47, 60

Enrico Fini, Victor G Turrisi Da Costa, Xavier Alameda-Pineda, Elisa Ricci, Karteek Alahari, and Julien Mairal. Self-supervised models are continual learners. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022. 22, 82

Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24(6):381–395, 1981. 20

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In Proceedings of the International Conference on Machine Learning (ICML), 2018. 4

Jean-Sébastien Franco and Edmond Boyer. Exact polyhedral visual hulls. In Proceedings of the British Machine Vision Conference (BMVC), 2003. 19

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In Proceedings of the International Conference on Learning Representations (ICLR), 2019. 3

Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022. 21, 74

Michael C Fu. Gradient estimation. Handbooks in operations research and management science, 13:575–616, 2006. 34

Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In Proceedings of the International Conference on Machine Learning (ICML), 2018. 8, 45, 61

Yasutaka Furukawa, Brian Curless, Steven M Seitz, and Richard Szeliski. Towards internet-scale multi-view stereo. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2010. 2, 3, 20

Valentin Gabeur, Jean-Sébastien Franco, Xavier Martin, Cordelia Schmid, and Gregory Rogez. Moulding humans: Non-parametric 3d human shape estimation from single images. In Proceedings of the International Conference on Computer Vision (ICCV), 2019. 19

Yuting Gao, Jia-Xin Zhuang, Shaohui Lin, Hao Cheng, Xing Sun, Ke Li, and Chunhua Shen. Disco: Remedy self-supervised learning on lightweight models with distilled contrastive learning. In Proceedings of the European Conference on Computer Vision (ECCV), 2022. 23, 47

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics, 2010. 2

Rahul Goel, Dhawal Sirikonda, Saurabh Saini, and P.J. Narayanan. Interactive segmentation of radiance fields. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023. 67

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. International Journal of Computer Vision (IJCV), 129:1789–1819, 2021. 46

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. arXiv preprint arXiv:1706.02677, 2017. 2

Will Grathwohl, Dami Choi, Yuhuai Wu, Geoffrey Roeder, and David Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient

estimation. In Proceedings of the International Conference on Learning Representations (ICLR), 2018. 4, 27, 29

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. In Advances in Neural Information Processing Systems (NeurIPS), 2020. 2, 17, 22

Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018. 19

Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In Proceedings of the International Conference on Learning Representations (ICLR), 2021. 38, 115

Jun Guo, Xiaojian Ma, Yue Fan, Huaping Liu, and Qing Li. Semantic gaussians: Open-vocabulary scene understanding with 3d gaussian splatting. arXiv preprint arXiv:2403.15624, 2024. 66

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In Proceedings of the International Conference on Learning Representations (ICLR), 2016. 3

R. Hartley and A. Zisserman. Multiple View Geometry in Computer Vision. Cambridge University Press, 2003. 20

Ryuichiro Hataya, Jan Zdenek, Kazuki Yoshizoe, and Hideki Nakayama. Faster AutoAugment: Learning augmentation strategies using backpropagation. In Proceedings of the European Conference on Computer Vision (ECCV), 2020. 4, 15, 29, 34, 114

Ryuichiro Hataya, Jan Zdenek, Kazuki Yoshizoe, and Hideki Nakayama. Meta approach to data augmentation optimization. In Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV), 2022. 4, 26, 29

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the International Conference on Computer Vision (ICCV), 2015. 2

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2016. 2, 14, 36

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020. 2, 16

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022. 2, 6, 17, 44

Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. ACM Transactions on Graphics (ToG), 37(6):1–15, 2018. 21

Alexander Hermans, Georgios Floros, and Bastian Leibe. Dense 3d semantic mapping of indoor scenes from rgb-d images. In Proceedings of the IEEE International Conference on Robotics and Automatics (ICRA), 2014. 9

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2015. 3, 7, 22, 46, 47, 49, 51, 125, 133

Daniel Ho, Eric Liang, Xi Chen, Ion Stoica, and Pieter Abbeel. Population based augmentation: Efficient learning of augmentation policy schedules. In Proceedings of the International Conference on Machine Learning (ICML), pages 2731–2741, 2019. 4, 15, 26, 29, 114

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In Proceedings of the International Conference on Machine Learning (ICML), 2019. 7, 81

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In Proceedings of the International Conference on Learning Representations (ICLR), 2022. 3, 7, 81

Wei Huang, Zhiliang Peng, Li Dong, Furu Wei, Jianbin Jiao, and Qixiang Ye. Generic-to-specific distillation of masked autoencoders. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023. 7, 8, 22, 23, 45, 47

Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, 2021. 10, 65, 73, 74

Sergey Ioffe. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International Conference on Machine Learning (ICML), 2015. 2

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018. 3, 81

Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2010. 14

Mengqi Ji, Juergen Gall, Haitian Zheng, Yebin Liu, and Lu Fang. Surfacenet: An end-to-end 3d neural network for multiview stereopsis. In Proceedings of the International Conference on Computer Vision (ICCV), 2017. 19

Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In Proceedings of the International Conference on Machine Learning (ICML), 2021. 18

Yingwenqi Jiang, Jiadong Tu, Yuan Liu, Xifeng Gao, Xiaoxiao Long, Wenping Wang, and Yuexin Ma. Gaussianshader: 3d gaussian splatting with shading functions for reflective surfaces. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024. 77, 80, 144

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT: Distilling BERT for natural language understanding. In Findings of the Association for Computational Linguistics: EMNLP, 2020. 7, 8, 22, 23, 45, 47

Kim Jun-Seong, GeonU Kim, Kim Yu-Ji, Yu-Chiang Frank Wang, Jaesung Choe, and Tae-Hyun Oh. Dr. splat: Directly referring 3d gaussian splatting via direct language embedding registration. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2025. 10, 66, 67, 79, 80, 84

Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. ACM Transactions on Graphics (ToG), 32(3):1–13, 2013. 19

Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In Proceedings of the fourth Eurographics symposium on Geometry processing, 2006. 19

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. In Proceedings of the ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques, 2023. 3, 9, 22, 23, 64, 66, 68, 74, 83, 84, 85

Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. Lerf: Language embedded radiance fields. In Proceedings of the International Conference on Computer Vision (ICCV), 2023. xiv, 9, 22, 23, 64, 66, 67, 70, 73, 75, 78, 79, 80, 84, 139, 142, 143, 150

Chung Min Kim, Mingxuan Wu, Justin Kerr, Ken Goldberg, Matthew Tancik, and Angjoo Kanazawa. Garfield: Group anything with radiance fields. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024. 10, 23, 66, 67

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Proceedings of the International Conference on Learning Representations (ICLR), 2015. 15

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In Proceedings of the International Conference on Computer Vision (ICCV), 2023. 3, 9, 10, 16, 64, 65, 66, 72, 74, 84, 135

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences, 114(13):3521–3526, 2017. 61

Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. ACM Transactions on Graphics (ToG), 36(4):1–13, 2017. 74

Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. Advances in Neural Information Processing Systems (NeurIPS), 2022. 9, 64, 66, 67, 84

Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete structures. In International Conference on Machine Learning (ICML), 2002. 10, 65, 70

Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. Point-based neural rendering with per-view optimization. In Computer Graphics Forum, volume 40(4), pages 29–43. Wiley Online Library, 2021. 22

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical Report Technical Report, University of Toronto, 2009. 26, 35

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems (NIPS), 2012. 2, 4, 14, 15

Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2006. 1, 13

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998. 2, 14

Byeonghyeon Lee, Howoong Lee, Xiangyu Sun, Usman Ali, and Eunbyung Park. Deblurring 3d gaussian splatting. In Proceedings of the European Conference on Computer Vision (ECCV), 2024. 80

Benjamin Lefaudeux, Francisco Massa, Diana Liskovich, Wenhan Xiong, Vittorio Caggiano, Sean Naren, Min Xu, Jieru Hu, Marta Tintore, Susan Zhang, Patrick Labatut, Daniel Haziza, Luca Wehrstedt, Jeremy Reizenstein, and Grigory Sizov. xformers: A modular and hackable transformer modelling library. https://github.com/facebookresearch/xformers, 2022. 83

Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In Proceedings of the European Conference on Computer Vision (ECCV), 2024. 85

M. Levoy and P. Hanrahan. Light field rendering. Proceedings of the ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques, 1996. 21

Boyi Li, Kilian Q Weinberger, Serge Belongie, Vladlen Koltun, and Rene Ranftl. Language-driven semantic segmentation. In Proceedings of the International Conference on Learning Representations (ICLR), 2022a. 66

Chun Hung Li and CK Lee. Minimum cross entropy thresholding. Pattern recognition, 26(4):617–625, 1993. 136

Chunyuan Li, Jianwei Yang, Pengchuan Zhang, Mei Gao, Bin Xiao, Xiyang Dai, Lu Yuan, and Jianfeng Gao. Efficient self-supervised vision transformers for representation learning. In Proceedings of the International Conference on Learning Representations (ICLR), 2022b. 6, 44

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In Proceedings of the International Conference on Machine Learning (ICML), 2022c. 3, 18

Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. Visualbert: A simple and performant baseline for vision and language. arXiv preprint arXiv:1908.03557, 2019. 18

Yonggang Li, Guosheng Hu, Yongtao Wang, Timothy M. Hospedales, Neil Martin Robertson, and Yongxin Yang. DADA: differentiable automatic data augmentation. In Proceedings of the European Conference on Computer Vision (ECCV), 2020. 4, 26, 29, 34, 37, 114, 118

Zhizhong Li and Derek Hoiem. Learning without forgetting. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 40(12):2935–2947, 2017. 22

Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast AutoAugment. In Advances in Neural Information Processing Systems (NeurIPS), 2019. 4, 26, 27, 29, 37, 114, 118

C. Lin, M. Guo, C. Li, X. Yuan, W. Wu, J. Yan, D. Lin, and W. Ouyang. Online hyper-parameter learning for auto-augmentation strategy. In Proceedings of the International Conference on Computer Vision (ICCV), 2019. 26

Aoming Liu, Zehao Huang, Zhiwu Huang, and Naiyan Wang. Direct differentiable augmentation search. In Proceedings of the International Conference on Computer Vision (ICCV), 2021. 15, 29, 34

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In Advances in Neural Information Processing Systems (NeurIPS), 2022a. 7, 81

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In Advances in Neural Information Processing Systems (NeurIPS), 2023a. 18

Kunhao Liu, Fangneng Zhan, Jiahui Zhang, Muyu Xu, Yingchen Yu, Abdulmotaleb El Saddik, Christian Theobalt, Eric Xing, and Shijian Lu. Weakly supervised 3d open-vocabulary segmentation. Advances in Neural Information Processing Systems (NeurIPS), 2023b. 9, 64, 66, 67, 70

Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In Proceedings of the International Conference on Computer Vision (ICCV), 2019. 3, 21

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022b. 6, 15

Matthew M Loper and Michael J Black. Opendr: An approximate differentiable renderer. In Proceedings of the European Conference on Computer Vision (ECCV), 2014. 3, 21

William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In Proceedings of the ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques, 1987. 19

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In Proceedings of the International Conference on Learning Representations (ICLR), 2017. 15

David G Lowe. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision (IJCV), 60:91–110, 2004. 1, 13

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In Advances in Neural Information Processing Systems (NeurIPS), 2019. 18

S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013. 52, 125

Hidenobu Matsuki, Riku Murai, Paul HJ Kelly, and Andrew J Davison. Gaussian splatting slam. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024. 22

John McCormac, Ankur Handa, Andrew Davison, and Stefan Leutenegger. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. In Proceedings of the IEEE International Conference on Robotics and Automatics (ICRA), 2017. 9

Sachin Mehta and Mohammad Rastegari. Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer. In Proceedings of the International Conference on Learning Representations (ICLR), 2022. 6

Marina Meila and Jianbo Shi. Learning segmentation by random walks. Advances in neural information processing systems (NIPS), 13, 2000. 10, 65

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019. 19

Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. ACM Transactions on Graphics (ToG), 38(4):1–14, 2019. 74

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the i, 65(1):99–106, 2021. 3, 9, 21, 23, 64, 65, 74, 83, 85

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. In Proceedings of the AAAI Conference on Artificial Intelligence, 2020. 8, 45, 60, 61

Ashkan Mirzaei, Tristan Aumentado-Armstrong, Konstantinos G Derpanis, Jonathan Kelly, Marcus A Brubaker, Igor Gilitschenski, and Alex Levinshtein. Spin-nerf: Multiview segmentation and perceptual inpainting with neural radiance fields. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 20669–20679, 2023. xiii, 77, 78, 143, 144

Samuel G. Müller and Frank Hutter. TrivialAugment: tuning-free yet state-of-the-art data augmentation. In Proceedings of the International Conference on Computer Vision (ICCV), 2021. 5, 26, 27, 29, 36, 37, 38, 54, 114, 115, 116, 127

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. In Proceedings of the ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques, 2022. 21

Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: A versatile and accurate monocular slam system. IEEE transactions on robotics, 31 (5):1147–1163, 2015. 21

K L Navaneet, Soroush Abbasi Koohpayegani, Ajinkya Tejankar, and Hamed Pirsiavash. Simreg: Regression as a simple yet effective tool for self-supervised knowledge distillation. In Proceedings of the British Machine Vision Conference (BMVC), 2021. 23, 47

Quang Nguyen, Truong Vu, Anh Tran, and Khoi Nguyen. Dataset diffusion: Diffusion-based synthetic dataset generation for pixel-level semantic segmentation. In Advances in Neural Information Processing Systems (NeurIPS), 2023. 48

David Nistér. An efficient solution to the five-point relative pose problem. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 26(6):756–770, 2004. 2, 20

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748, 2018. 16

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. Transactions on Machine Learning Research (TMLR), 2024. xi, xiv, 3, 6, 7, 10, 18, 22, 23, 44, 45, 47, 53, 54, 57, 60, 61, 64, 65, 68, 70, 72, 82, 84, 127, 130, 136, 137

Nobuyuki Otsu et al. A threshold selection method from gray-level histograms. Automatica, 11(285-296):23–27, 1975. 140, 141

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019. 19

Mattis Paulin, Jérôme Revaud, Zaid Harchaoui, Florent Perronnin, and Cordelia Schmid. Transformation pursuit for image classification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2014. 28, 37

Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In Proceedings of the International Conference on Computer Vision (ICCV), 2019. 26, 35, 52, 125, 133, 134

Florent Perronnin and Christopher Dance. Fisher kernels on visual vocabularies for image categorization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2007. 1, 14

Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In Proceedings of the European Conference on Computer Vision (ECCV), 2010. 1, 14

Hanspeter Pfister, Matthias Zwicker, Jeroen Van Baar, and Markus Gross. Surfels: Surface elements as rendering primitives. In Proceedings of the ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques, 2000. 19

Justin Pinkney. Image mixer, 2022. Lambda Labs. 44, 48, 51, 52, 53, 58, 59, 129, 134

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2017a. 19

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Advances in Neural Information Processing Systems (NeurIPS), 2017b. 20

Minghan Qin, Wanhua Li, Jiawei Zhou, Haoqian Wang, and Hanspeter Pfister. Langsplat: 3d language gaussian splatting. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024. xiii, xiv, 10, 22, 23, 66, 67, 75, 78, 79, 139, 140, 141, 142, 143, 150

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Proceedings of the International Conference on Machine Learning (ICML), 2021. 3, 9, 18, 64, 66, 68, 84

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. arXiv preprint arXiv:2408.00714, 2024. 10, 65, 72, 74, 135

Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In Proceedings of the International Conference on Computer Vision (ICCV), 2021. 77

Zhongzheng Ren, Aseem Agarwala, Bryan Russell, Alexander G Schwing, and Oliver Wang. Neural volumetric object selection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022. xi, xiii, 74, 77, 78, 143, 144, 145, 146

Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2017. 19

R Tyrrell Rockafellar. Monotone operators and the proximal point algorithm. SIAM journal on control and optimization, 14(5):877–898, 1976. 28

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022. 8, 15, 44, 48, 51

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In Proceedings of the International Conference on Learning Representations (ICLR), 2015. 47

Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In Proceedings of the International Conference on Computer Vision (ICCV), 2011. 1

Szymon Rusinkiewicz and Marc Levoy. Qsplat: A multiresolution point rendering system for large meshes. In Proceedings of the ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques, 2000. 19, 22

Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. arXiv preprint arXiv:1911.08731, 2019. 61

Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In Proceedings of the ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques, 2022a. 16, 48

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In Advances in Neural Information Processing Systems (NeurIPS), 2022b. 44

Mert Bulent Sariyildiz, Julien Perez, and Diane Larlus. Learning visual representations with caption annotations. In Proceedings of the European Conference on Computer Vision (ECCV), 2020. 18

Mert Bülent Sarıyıldız, Karteek Alahari, Diane Larlus, and Yannis Kalantidis. Fake it till you make it: Learning transferable representations from synthetic imagenet clones. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023. 48, 58

Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In Proceedings of the International Conference on Machine Learning (ICML), 2021. 83

Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2016. 2, 3, 20

John Schulman, Filip Wolski, Prafulla Dhariwal, and Oleg Radford, Alec a^{2}nd Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017. 28, 35

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2014. 60

Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. IEEE Transactions on pattern analysis and machine intelligence (PAMI), 22(8):888–905, 2000. 10, 65, 70

Jin-Chuan Shi, Miao Wang, Hao-Bin Duan, and Shao-Hua Guan. Language embedded 3d gaussians for open-vocabulary scene understanding. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024. 67

Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. Journal of big data, 6(1):1–48, 2019. 4

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Proceedings of the International Conference on Learning Representations (ICLR), 2015. 2, 14

Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. Pacific Journal of Mathematics, 21(2):343–348, 1967. 17

Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In Proceedings of the International Conference on Computer Vision (ICCV), 2003. 1, 13, 14

Alexander J Smola and Risi Kondor. Kernels and regularization on graphs. In Learning Theory and Kernel Machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop, 2003. 10, 65

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1):1929–1958, 2014. 2

Samuel Stanton, Pavel Izmailov, Polina Kirichenko, Alexander A Alemi, and Andrew G Wilson. Does knowledge distillation really work? In Advances in Neural Information Processing Systems (NeurIPS), 2021. 8, 49

Quan Sun, Yuxin Fang, Ledell Wu, Xinlong Wang, and Yue Cao. Eva-clip: Improved training techniques for clip at scale. arXiv preprint arXiv:2303.15389, 2023. xiv, 45, 125, 131, 132

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for BERT model compression. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2019. 8, 22, 23, 45, 47

Teppei Suzuki. Teachaugment: Data augmentation optimization using teacher knowledge. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022. 37

Hao Tan and Mohit Bansal. Lxmert: Learning cross-modality encoder representations from transformers. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, 2019. 18

Sebastian Thrun. Probabilistic robotics. Communications of the ACM, 45(3):52–57, 2002. 21

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. arXiv preprint arXiv:1906.05849, 2019. 35

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. In Proceedings of the International Conference on Learning Representations (ICLR), 2020. 47, 133

Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: a factorization method. International Journal of Computer Vision (IJCV), 9:137–154, 1992. 2, 20

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In Proceedings of the International Conference on Machine Learning (ICML), 2021. 8, 16, 22, 23, 45, 47, 133

Brandon Trabucco, Kyle Doherty, Max Gurinas, and Ruslan Salakhutdinov. Effective data augmentation with diffusion models. arXiv preprint arXiv:2302.07944, 2023. 8, 16, 44, 48, 82

Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. In International Workshop on Vision Algorithms, 1999. 20

Vadim Tschernezki, Iro Laina, Diane Larlus, and Andrea Vedaldi. Neural feature fusion fields: 3d distillation of self-supervised 2d image representations. In Proceedings of the International Conference on 3D Vision (3DV), 2022. xii, 9, 65, 67, 70, 146, 148

Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. In Proceedings of the International Conference on Computer Vision (ICCV), 2019. 47

Shimon Ullman. The interpretation of visual motion. Massachusetts Inst of Technology Pr, 1979. 20

Gido M Van de Ven, Tinne Tuytelaars, and Andreas S Tolias. Three types of incremental learning. Nature Machine Intelligence, 4(12):1185–1197, 2022. 22

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in Neural Information Processing Systems (NeurIPS), 2017. 2, 14, 83

Riccardo Volpi and Vittorio Murino. Addressing model vulnerability to distributional shifts over image transformation sets. In Proceedings of the International Conference on Computer Vision (ICCV), pages 7980–7989, 2019. 28

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 52, 125

Chieh-Chih Wang, Charles Thorpe, Sebastian Thrun, Martial Hebert, and Hugh Durrant-Whyte. Simultaneous localization, mapping and moving object tracking. The International Journal of Robotics Research, 26(9):889–916, 2007. 21

Huan Wang, Suhas Lohit, Michael N Jones, and Yun Fu. What makes a "good" data augmentation in knowledge distillation—a statistical perspective. In Advances in Neural Information Processing Systems (NeurIPS), 2022. 8, 49

Junjie Wang, Jiemin Fang, Xiaopeng Zhang, Lingxi Xie, and Qi Tian. Gaussianeditor: Editing 3d gaussians delicately with text instructions. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024a. 23, 66

Lin Wang and Kuk-Jin Yoon. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 44(6):3048–3068, 2021. 46, 48

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 2024b. 82

Mengzhao Wang, Xiaoliang Xu, Qiang Yue, and Yuxiang Wang. A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search. arXiv preprint arXiv:2101.12631, 2021a. 142

Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In Proceedings of the European Conference on Computer Vision (ECCV), 2018. 19

Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. In Proceedings of the ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques, 2017. 19

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024c. 20, 85

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In Advances in Neural Information Processing Systems (NeurIPS), 2020. 8, 45

Xiaoxing Wang, Xiangxiang Chu, Junchi Yan, and Xiaokang Yang. DAAS: Differentiable architecture and augmentation policy search. arXiv preprint arXiv:2109.15273, 2021b. 26, 29, 34

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning, 8(3):229–256, 1992. 4, 27, 29

Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. Convnext v2: Co-designing and scaling convnets with

masked autoencoders. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023. 6, 15

Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024a. 22

Kan Wu, Jinnian Zhang, Houwen Peng, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. TinyViT: Fast pretraining distillation for small vision transformers. In Proceedings of the European Conference on Computer Vision (ECCV), 2022. 22, 23, 47, 60, 82

Yanmin Wu, Jiarui Meng, Haijie Li, Chenming Wu, Yahao Shi, Xinhua Cheng, Chen Zhao, Haocheng Feng, Errui Ding, Jingdong Wang, et al. Opengaussian: Towards point-level 3d gaussian-based open vocabulary understanding. In Advances in Neural Information Processing Systems (NeurIPS), 2024b. 10, 64, 66, 67, 79, 80, 84

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2015. 19

Monika Wysoczańska, Oriane Siméoni, Michaël Ramamonjisoa, Andrei Bursuc, Tomasz Trzciński, and Patrick Pérez. Clip-dinoiser: Teaching clip a few dino tricks for open-vocabulary semantic segmentation. In Proceedings of the European Conference on Computer Vision (ECCV), 2024. 70

Haohang Xu, Jiemin Fang, Xiaopeng Zhang, Lingxi Xie, Xinggang Wang, Wenrui Dai, Hongkai Xiong, and Qi Tian. Bag of instances aggregation boosts self-supervised distillation. In Proceedings of the International Conference on Learning Representations (ICLR), 2022. 23, 47, 60

Lihe Yang, Xiaogang Xu, Bingyi Kang, Yinghuan Shi, and Hengshuang Zhao. Freemask: Synthetic images with dense annotations make stronger segmentation models. In Advances in Neural Information Processing Systems (NeurIPS), 2023. 16, 48

Ziyi Yang, Xinyu Gao, Yangtian Sun, Yihua Huang, Xiaoyang Lyu, Wen Zhou, Shaohui Jiao, Xiaojuan Qi, and Xiaogang Jin. Spec-gaussian: Anisotropic view-dependent appearance for 3d gaussian splatting. arXiv preprint arXiv:2402.15870, 2024. 80

Jianglong Ye, Naiyan Wang, and Xiaolong Wang. Featurenerf: Learning generalizable nerfs by distilling foundation models. In Proceedings of the International Conference on Computer Vision (ICCV), 2023. 23, 66, 67

Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian grouping: Segment and edit anything in 3d scenes. In Proceedings of the European Conference on Computer Vision (ECCV), 2024. 9, 10, 23, 64, 66, 67, 84

Lin Yen-Chen, Pete Florence, Jonathan T Barron, Tsung-Yi Lin, Alberto Rodriguez, and Phillip Isola. Nerf-supervision: Learning dense object descriptors from neural radiance fields. In Proceedings of the International Conference on Learning Representations (ICLR), 2022. 67, 74, 77

Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. ACM Transactions On Graphics (TOG), 38(6):1–14, 2019. 22

Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2017. 47

Haiyang Ying, Yixuan Yin, Jinzhi Zhang, Fan Wang, Tao Yu, Ruqi Huang, and Lu Fang. Omniseg3d: Omniversal 3d segmentation via hierarchical contrastive learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024. 9, 64, 66, 67, 77

Li Yuan, Francis EH Tay, Guilin Li, Tao Wang, and Jiashi Feng. Revisiting knowledge distillation via label smoothing regularization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020. 8, 45, 61

Yuanwen Yue, Anurag Das, Francis Engelmann, Siyu Tang, and Jan Eric Lenssen. Improving 2d feature representations by 3d-aware fine-tuning. In Proceedings of the European Conference on Computer Vision (ECCV), 2024. 23, 66

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith, editors, Proceedings of the British Machine Vision Conference (BMVC), pages 87.1–87.12. BMVA Press, September 2016. ISBN 1-901725-59-6. doi: 10.5244/C.30.87. 36

Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In Proceedings of the International Conference on Learning Representations (ICLR), 2017. 47

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In Proceedings of the International Conference on Learning Representations (ICLR), 2018. 15, 53

Xinyu Zhang, Qiang Wang, Jian Zhang, and Zhao Zhong. Adversarial AutoAugment. In Proceedings of the International Conference on Learning Representations (ICLR), 2020. 4, 29

Borui Zhao, Quan Cui, Renjie Song, Yiyu Qiu, and Jiajun Liang. Decoupled knowledge distillation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022. 47, 133

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In Proceedings of the International Conference on Computer Vision (ICCV), 2021. 20

Lingzhe Zhao, Peng Wang, and Peidong Liu. Bad-gaussians: Bundle adjusted deblur gaussian splatting. In Proceedings of the European Conference on Computer Vision (ECCV), 2024. 80

Yu Zheng, Zhi Zhang, Shen Yan, and Mi Zhang. Deep AutoAugmentation. In Proceedings of the International Conference on Learning Representations (ICLR), 2022. 26, 29, 37, 114, 118

Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2017. 7, 52, 125, 133, 134

Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. ibot: Image bert pre-training with online tokenizer. In Proceedings of the International Conference on Learning Representations (ICLR), 2022a. 17, 18

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. International Journal of Computer Vision (IJCV), 130 (9):2337–2348, 2022b. 44, 48

Shijie Zhou, Haoran Chang, Sicheng Jiang, Zhiwen Fan, Zehao Zhu, Dejia Xu, Pradyumna Chari, Suya You, Zhangyang Wang, and Achuta Kadambi. Feature 3dgs: Supercharging 3d gaussian splatting to enable distilled feature fields. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024. 10, 23, 66, 67

Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. In Proceedings of the International Conference on Learning Representations (ICLR), Workshop Track, 2018. 3, 81

B Zoph. Neural architecture search with reinforcement learning. In Proceedings of the International Conference on Learning Representations (ICLR), 2016. 4

Xingxing Zuo, Pouya Samangouei, Yunwen Zhou, Yan Di, and Mingyang Li. Fmgs: Foundation model embedded 3d gaussian splatting for holistic 3d scene understanding. arXiv preprint arXiv:2401.01970, 2024. 9, 10, 23, 64, 66, 67, 70, 73, 80, 142

Appendix A

Appendix - SLACK: Stable Learning of Augmentations with Cold-start and KL regularization

In this supplementary, we first give a short overview of prior work's key aspects (Section A.1), then we provide additional details about the experimental protocol that was used to obtain our results (Section A.2) and analyse in more details the results reported in the paper and the impact of our approach on the stability of the search process (Section A.3).

A.1 Summary of prior art

Overall, methods for augmentation search mostly differ in key design choices that are highlighted in Table A.1.

A.2 Experimental setup

In this section, we first describe the practical implementation of our gradient estimation, then we compare our magnitude ranges to those of other methods, and finally we describe in more detail our search and evaluation protocols.

A.2.1 Gradient estimation

In this section, we describe the practical implementation of the gradient estimates derived in our method section.

To optimize the augmentation policies, we minimize an approximation to the upper-level objective $\mathcal{F}(\phi) := \mathcal{L}_{\text{val}}(\theta^\star(\phi))$ defined as $\hat{\mathcal{F}}(\phi) := \mathcal{L}_{\text{val}}(\hat{\theta}(\phi))$, where we replaced the intractable lower-level solution $\theta^\star(\phi)$ by an approximate solution $\hat{\theta}(\phi)$. Such approximate solution is obtained by performing one gradient step to optimize the lower-level objective starting from the current parameter θ, i.e. $\hat{\theta}(\phi) := \theta - \eta \nabla_\theta \mathcal{L}_{\text{train}}(\theta, \phi)$. The

Method	Optimization	Prior-free*	Full set	Search time
AA Cubuk et al. (2019)	Reinfor. learning	✗	✗	5000
PBA Ho et al. (2019)	Population-based	✗	✗	5
Fast AA Lim et al. (2019)	Bayesian optim.	✗	✗	3.5
Faster AA Hataya et al. (2020)	RELAX	✗	✗	0.23
DADA Li et al. (2020)	RELAX	✗	✗	0.1
RA Cubuk et al. (2020)	Exhaustive search	✗	✓	25
TrivialA Müller and Hutter (2021)	No optimization	✗	✓	NA
DeepAA Zheng et al. (2022)	Greedy algorithm	✓**	✓***	9
SLACK(Ours)	REINFORCE	✓	✓	4

* Prior-free refers to methods that do not use any default transformations.
** DeepAA does not use default transformations during search but during pretraining.
*** DeepAA pretrains on a reduced dataset.

TABLE A.1: **Key aspects of automatic data augmentation methods.** Our approach, SLACK, tackles the corresponding bilevel optimization problem using the REINFORCE gradient estimator. It is prior-free (i.e. does not rely on default transformations) and can use the full training set while maintaining a reasonable search time (indicated in GPU hours, for search on CIFAR with WRN-40-2).

gradient $\nabla_\phi \mathcal{F}$ is then naturally approximated by $\nabla_\phi \hat{\mathcal{F}}$ which is computed by applying the chain rule:

$$\nabla_\phi \mathcal{F} \approx \nabla_\phi \hat{\mathcal{F}} = \nabla_\theta \mathcal{L}_{\text{val}}(\hat{\theta}(\phi))^\top \nabla_\phi \hat{\theta}(\phi).$$

The Jacobian $\nabla_\phi \hat{\mathcal{F}}$ can be computed explicitly using the Score method which yields:

$$\nabla_\phi \hat{\theta}(\phi) = -\eta \mathbb{E}_{\tau \sim p_\phi} \left[\nabla_\theta \ell_{\text{train}}(\theta, \tau) \nabla_\phi \log p_\phi(\tau)^\top \right].$$

In practice, expectations over the data and augmentation policies are estimated with batches. At a given iteration, we sample B_{aug} augmentations from p_ϕ and then apply each of them to a batch of training data B_{train} from $\mathcal{D}_{\text{train}}$ to approximate $\nabla_\phi \hat{\theta}(\phi)$. Finally, we use a batch B_{val} of data from $\mathcal{D}_{\text{val}}$ to estimate the validation loss. Denoting N_a, N_t, N_v the size of the augmentation, training and validation batches respectively, and

$$\hat{l}_{\text{val}}(\theta) := \frac{1}{N_v} \sum_{(x,y) \in B_{\text{val}}} \left[\ell(y, f_\theta(x)) \right],$$

$$\hat{l}_{\text{train}}(\theta, \tau) := \frac{1}{N_t} \sum_{(x,y) \in B_{\text{train}}} \left[\ell(y, f_\theta(\tau(x))) \right],$$

our gradient estimate can be expressed as

$$\begin{aligned}\nabla_\phi \mathcal{F} &\approx -\frac{\eta}{N_a}\nabla_{\hat{\theta}}\hat{l}_{\text{val}}(\theta)\left(\sum_{\tau\in B_{\text{aug}}}\nabla_\theta \hat{l}_{\text{train}}(\theta,\tau)\nabla_\phi \log p_\phi(\tau)^T\right)\\ &= -\frac{\eta}{N_a}\sum_{\tau\in B_{\text{aug}}}\left(\nabla_\theta \hat{l}_{\text{val}}(\theta)^T\nabla_\theta \hat{l}_{\text{train}}(\theta,\tau)\right)\nabla_\phi \log p_\phi(\tau)\end{aligned}$$

In other words, the upper-level gradient is a weighted sum of the scores $\nabla_\phi \log p_\phi(\tau)$, with the weights representing the alignment between the gradients of i) the loss on the training data transformed with τ (evaluated at θ), and ii) the loss on the validation data (evaluated at $\hat{\theta}$, i.e. one step ahead).

In practice, the lower-level learning rate decreases with a cosine schedule. As we do not want our upper-level gradient updates to shrink, we set η to the initial value of the lower-level learning rate instead of its current value.

A.2.2 Magnitude ranges.

The ranges used for mapping the magnitudes to [0,1] vary across methods; we indicate this mapping for each method in Table A.2. For transformations with respect to which the datasets naturally exhibit symmetries (Shear, Translate, Rotate, Enhance), once we have sampled a magnitude, we randomly select a direction. Note that SLACK's ranges are larger than the usual ones (i.e. those of TA (RA)), which gives more flexibility during the optimization of our magnitude upper-bounds μ. The latter is initialized at 0.75. Experimentally, we noted that this initialization should be high enough to favour exploration and avoid over-fitting during pre-training. We observed that any initialization in the $[0.75, 0.9]$ range consistently works well across datasets.

A.2.3 Image pre-processing

Table A.3 indicates the image pre-processing choices on ImageNet-100 and DomainNet for TrivialAugment (Müller and Hutter, 2021), DomainBed (Gulrajani and Lopez-Paz, 2021) and SLACK.

ImageNet-100 and DomainNet images have variable original sizes. In the literature, training images are commonly resized with RandomResizeCrop. For testing, TrivialAugment uses Resize(256)+CenterCrop((224,224)), preserving the aspect ratio, while DomainBed directly applies Resize((224,224)), degrading the aspect ratio but preserving the image content. For each method, we stick to the authors' choices, as we

Application	Transformation	Method			
		TA (RA)	TA (Wide)	DomainBed	Ours
Sampled	ShearX/Y	$[0, 0.3]$	$[0, 0.99]$	-	$[0, 1]$
	Translate X/Y	$[0, 0.45]^*$	$[0, 32\text{px}]^{**}$	-	$[0, 0.75]$
	Rotate	$[0, 30]$	$[0, 135]$	-	$[0, 90]$
	Posterize	$[4, 8]$	$[2, 8]$	-	$[2, 8]$
	Solarize	$[0, 255]$	$[0, 255]$	-	$[0, 255]$
	Enhance***	$[0, 0.9]$	$[0, 0.99]$	-	$[0, 0.99]$
	Cutout	$[0, 0.2]$	$[0, 0.6]$	-	$[0, 1]$
	RandCrop	-	-	-	$[0, 0.5]$
	RandResizeCrop	-	-	-	$[0.05, 1]$
Default ImageNet/DomainNet	ColorJitter****	$[0, 0.4]$	$[0, 0.4]$	$[0, 0.3]$	-
	RandResizeCrop	$[0.08, 1]$	$[0.08, 1]$	$[0.7, 1]$	-
Default CIFAR	Cutout	0.5	0.5	NA	-
	RandCrop	0.125	0.125	NA	-

TABLE A.2: Our magnitude ranges compared to those used by other methods.
* TrivialAugment (Müller and Hutter, 2021) uses $[-0.31, 0.31]$
** TrivialAugment sets the upper-bounds in pixels, not in proportion
*** Color, Contrast, Brightness, Sharpness **** Color, Contrast, Brightness

Dataset	Model	Train	Test
ImageNet-100	TrivialAugment	RRC((224,224))	R(256)+CC((224,224))
	SLACK	R(256)+RC((224,224))	R(256)+CC((224,224))
DomainNet	TrivialAugment ImageNet	RRC(224,224)	R(256)+CC((224,224))
	TrivialAugment CIFAR	R(256)+RC((224,224),padding=28)	R(256)+CC((224,224))
	DomainBed	RRC((224,224))	R((224,224))
	SLACK (Clipart, Sketch, Quickdraw)	R((224,224))	R((224,224))
	SLACK (Painting, Infograph, Real)	R(256)+RC((224,224))	R(256)+CC((224,224))

TABLE A.3: Image pre-processing on ImageNet-100 and DomainNet.
R: Resize, RC: RandomCrop, CC: CenterCrop, RRC: RandomResizedCrop.

experimentally noted that they yield the best results (e.g. using TrivialAugment's pre-processing for DomainBed degrades the performance, and vice-versa).

For SLACK, which does not apply RandomResizeCrop by default, we preprocess the training data and validation/testing data in the same way. For training, random cropping is applied instead of center cropping to fully exploit the data. For ImageNet-100, we use TrivialAugment's pre-processing. For DomainNet, we select the pre-processing strategy by cross-validation after pre-training.

A.2.4 Policy search

Hyperparameters used for policy search are indicated in Table A.4. They are chosen to satisfy two criteria that we found to be useful for obtaining a successful policy search: i) the validation loss after re-training should be similar (experimentally, slightly lower) to the one obtained after pre-training, and ii) the probability distributions should

Dataset	Network	Re-train iter	Unrolled iter	Batch size	Lower lr	Upper lr	KL weight × Upper lr
CIFAR10/100	WRN-40-2/WRN-28-10	1000	400	8×128	0.4 / 0.1	1	0.02
ImageNet-100	ResNet-18	2000	800	8×256	0.1	0.5	0.005
DomainNet	ResNet-18	800 - 1200	400	8×128	0.1	0.625 - 1.25	0.01

TABLE A.4: Hyperparameters used for search on CIFAR, ImageNet-100 and DomainNet.

Dataset	Network	Epochs	Batch size	Learning rate	Weight decay
CIFAR10/100	WRN-40-2, WRN-28x10	200	128	0.1	0.0005*
ImageNet-100	ResNet-18	270	256	0.1	0.001
DomainNet	ResNet-18	200	128	0.1	0.001

TABLE A.5: Hyperparameters used for training on CIFAR, ImageNet-100 and DomainNet. *: 0.0002 for CIFAR10 on WRN-40-2

vary at the same speed for all datasets. Our learning rate is 4 times larger for re-training on CIFAR10 than on CIFAR100. We observed that gradients on CIFAR10 are 4 times smaller in norm than those on CIFAR100, and that re-scaling the updates allows satisfying i) and ii) empirically. For DomainNet, we adapt the number of re-training steps to the dataset size. A fixed lower-level learning rate for all datasets experimentally satisfies i). We observed that the lower-level gradients differ in scale for each dataset. Satisfying ii) requires re-scaling the KL regularisation and accordingly changing the upper-level learning rate (so that KL weight $\times$ upper-level lr is constant).

The upper-level learning rate indicated in the Tables is the one used for updating π. We divide it by 40 for the optimization of μ to ensure slower updates for the magnitude parameter which we found to be sensitive to variations (or by 10 for ablations removing the KL regularization).

A.2.5 Policy evaluation

The hyperparameters used for the evaluation phase are indicated in Table A.5. For CIFAR10 and CIFAR100, we use the same hyperparameters as prior work.

A.3 Extended analysis

In this section, we further analyse the results of our search algorithm. We first illustrate the policies learned for all datasets. We then study in more detail the impact of our multi-stage approach with KL regularization, comparing it with single-stage (unrolled) and unregularized approaches and illustrating their instability.

Model	Magnitude model	CIFAR10		CIFAR100	
		WRN-40-2	WRN-28-10	WRN-40-2	WRN-28-10
FastAA/DADA initialization	Theirs	96.22	97.08	78.26	82.17
	Uniform	96.37	97.25	79.10	82.80
FastAA, reported	Original	96.4	97.3	79.3	82.7
FastAA, reproduced (evaluation only)	Original	96.4	97.22	79.11	82.82
	Uniform	96.37	97.30	79.15	82.84
DADA, reported	Original	96.4	97.3	79.1	82.5
DADA, reproduced (evaluation only)	Original	96.33	97.19	79.07	82.05
	Uniform	96.37	97.35	78.97	82.57
DeepAA, reported	Original	-	97.56	-	84.02
DeepAA, reproduced (evaluation only)	Original	96.46	97.48	79.62	83.85
	Uniform	96.55	97.47	78.89	83.62

TABLE A.6: **Why learning the magnitude range?** CIFAR10/100 accuracies for DADA, FastAA and DeepAA, when using their original magnitude model, or a simpler one which samples in a uniform manner in their ranges.

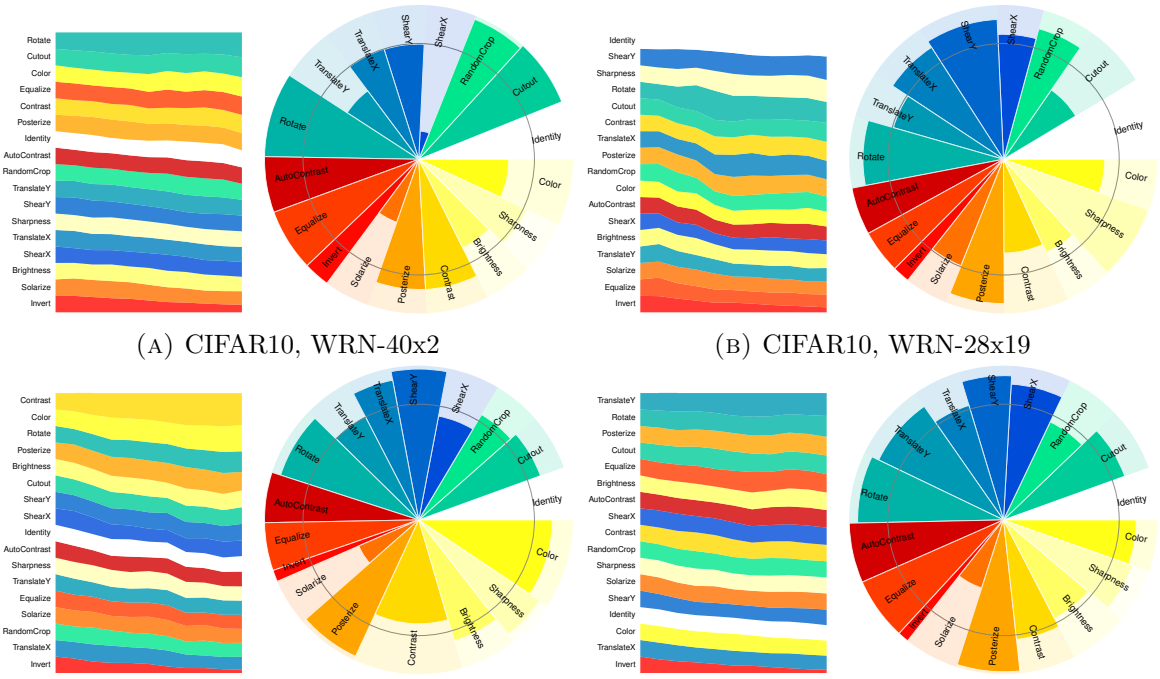
A.3.1 Uniform distribution: ablations on prior work

In this section, we motivate our choice of a uniform magnitude distribution, showing that it globally outperforms optimized magnitude models in prior work. To this end, we directly evaluate the policies provided by the authors without re-running their search procedure. We compare their learned magnitude model with a simpler one that consists in sampling the magnitudes uniformly on their $[0, 1]$-mapped ranges. We study three baselines: DADA (Li et al., 2020), FastAA (Lim et al., 2019) and DeepAA (Zheng et al., 2022). Note that their policy results from a search on CIFAR10, that they also use when evaluating on CIFAR100. Results are reported in Table A.6.

Parametrization. DADA and FastAA directly optimize a probability distribution over the set of all possible composite transformations (sub-policies) and lean a single magnitude value for each transformation in a sub-policy. They keep the top-k sub-policies for evaluation. DeepAA learns to compose transformations in a greedy manner and discretizes the magnitude ranges, learning a probability for each magnitude. We compare these learned magnitude values (FastAA, DADA) or learned probabilities (DeepAA) to our approach based on a uniform sampling.

DADA/FastAA. Our approach compares favorably to DADA's and FastAA's optimized models. We also compare both approaches on their initial policy (equal probabilities for all sub-policies, magnitudes set at mid-range). With uniform magnitude sampling, their initial policy (sampling among all possible sub-policies) performs similarly if not better than their optimized one (sampling among their top-k sub-policies).

DeepAA. Results on the policy provided by DeepAA are more nuanced: using uniform



(A) CIFAR10, WRN-40x2 (B) CIFAR10, WRN-28x19

(C) CIFAR100, WRN-40x2 (D) CIFAR100, WRN-28x10

FIGURE A.1: Illustrations of best policies found for CIFAR10 and CIFAR100 on WideResNet-40x2 and WideResNet-28x10 architectures. For each dataset and architecture, we show the evolution of the probability distribution π as training progresses (left) and the final learned policy as a pie chart (right), where slice widths represent π and and slice radii represent μ.

sampling improves results on CIFAR10 (on which their search was conduced) and degrades them on CIFAR100.

A.3.2 Visualization of the learned policies

CIFAR. The evolution of probability distributions for CIFAR10 and CIFAR100 and pie charts of the final policies are illustrated in Fig. A.1. It can be noted that Invert and Solarize, known to be detrimental, are systematically discarded. The policies learned are quite diverse, with different leading transformations for each distribution but global predominance of some transformations such as Cutout or Rotate. It can also be noted that magnitudes upper-bounds are in average higher for the larger WideResNet-28x10 networks: a larger learning capacity benefits more from harder transformations.

ImageNet-100. The best policy found for ImageNet-100 is illustrated in Fig. A.2. Interestingly, RandomResizeCrop is ranked quite low, yet our policy yields results comparable to TrivialAugment's (with a 86.18 average accuracy on this split), suggesting that other geometrical transformations such as Cutout, Rotate and ShearY are equivalently beneficial for training on ImageNet-100. We can note rather high magnitudes

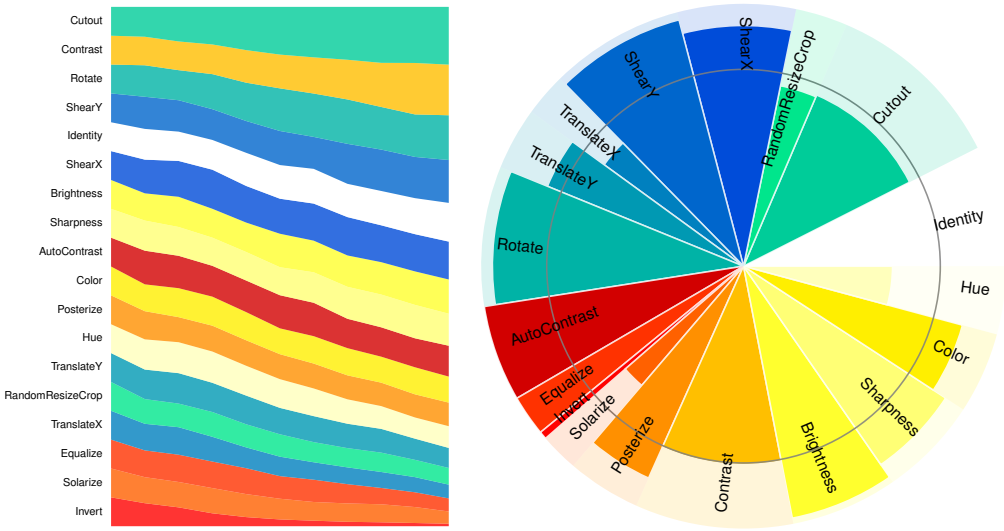


FIGURE A.2: ImageNet-100 policy on the best search split

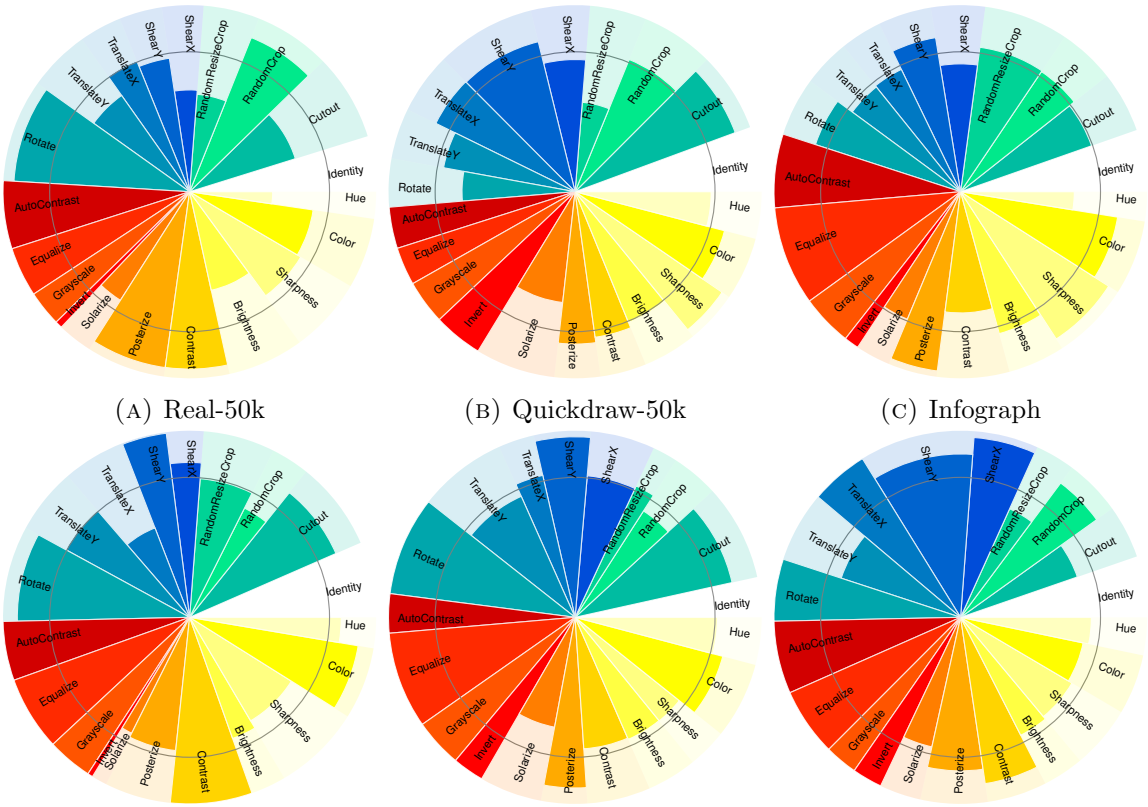
upper-bounds for the color jittering transformation TrivialAugment also applies by default (Color, Contrast, Brightness), which is consistent with the higher performance of TrivialAugment's (Wide) version compared to (RA).

DomainNet. The policies found by SLACK on DomainNet are illustrated in the pie charts Fig. A.3 for all domains. Some similarities with policies found on CIFAR and ImageNet can be noted. In particular, Invert and Solarize (that only inverts part of the pixels) are systematically discarded for all domains except Quickdraw. Invert is manually removed from TrivialAugment's baseline as it is known to be detrimental, and this seems to generalize to other domains. Also, Rotate and Cutout are globally favoured, similarly to the policies found on CIFAR and ImageNet-100.

However some differences mark specificities to each domain: i) on the strength of the transformations: for example, geometrical transformations are given high magnitudes on Clipart and lower ones on Real, ii) on their probabilities: color jittering transformations used for real images are globally assigned a high probability for Real, Painting and Infograph domains, and a much lower one for Clipart, which suggests that changes in color, contrast or brightness are less meaningful for this domain.

A.3.3 Avoiding instabilities with SLACK

In this section, we study how our augmentation policies evolve when removing the KL regularization or when using a single optimization stage instead of multiple ones, which corresponds to standard unrolled optimization. Evaluations in both settings are reported in Table A.7.



(A) Real-50k (B) Quickdraw-50k (C) Infograph

(D) Painting (E) Sketch (F) Clipart

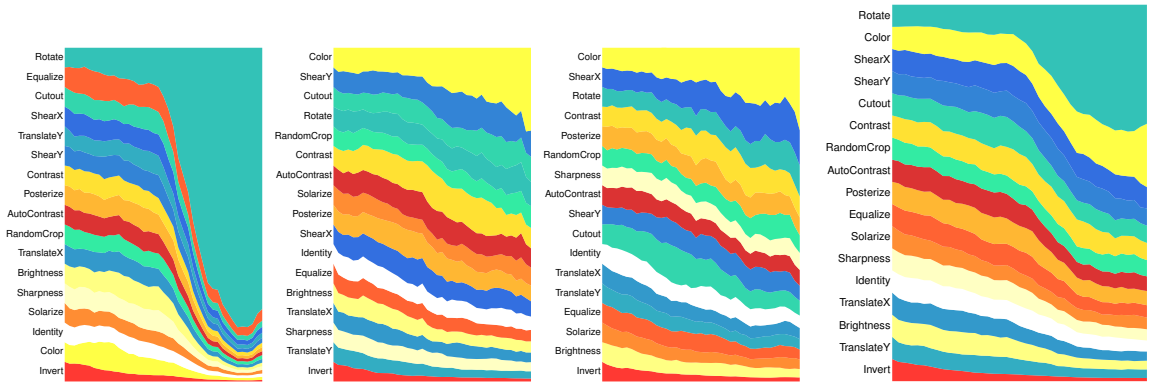
FIGURE A.3: Policies found on DomainNet. The three distributions from π forming the composite transformation are averaged.

				CIFAR10		CIFAR100	
SLACKvariant	Upper-level iter	Upper-level lr	KL weight	WRN-40-2	WRN-28-10	WRN-40-2	WRN-28-10
Unrolled w/ KL (Fig. A.5)	10000	0.25	0.005	96.30 ± .08	97.43 ± .04	79.54 ± .20	84.11 ± .13
SLACKw/o KL (Fig. A.6)	10×400	0.25	0	96.27 ± .05	97.06 ± .11	79.61 ± .13	83.79 ± .19
SLACK(Fig. A.1)	10×400	1	0.02	96.29 ± .08	97.46 ± .06	79.87 ± .11	84.08 ± .16

TABLE A.7: CIFAR10/100 accuracy with unregularized and single-stage approaches.

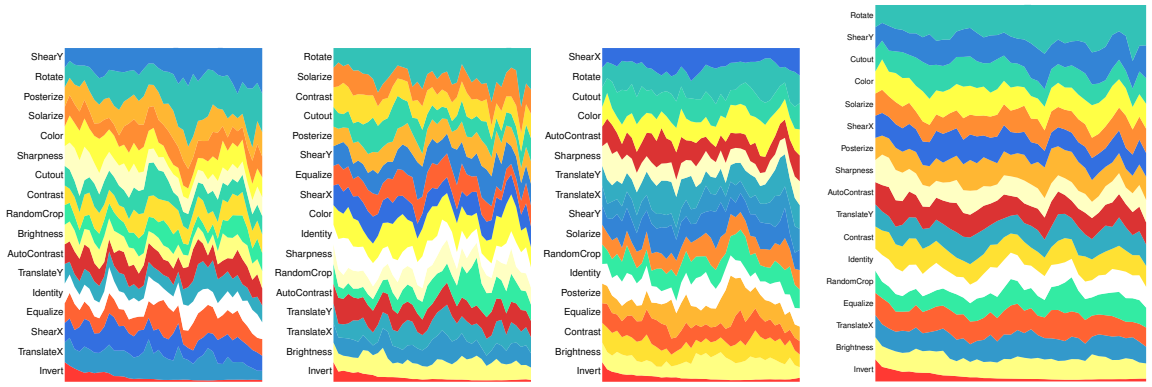
We first show that unrolled optimization is globally unstable and easily collapses, justifying the need for a regularization. We illustrate how entropy regularization prevents collapse and yields competitive results, but at the cost of high 'local' instability. These instabilities make the final performance highly dependent on the choice of some hyperparameters, such as the learning rate. The necessity to overcome these instabilities motivates our multi-stage procedure with an adaptive anchoring for the regularization. Lastly, we show that unregularized multi-stage optimization, while more stable than unregularized unrolled optimization, does not yield competitive results, confirming again the benefits of our KL regularization.

Unregularized unrolled optimization. Unrolled optimization is subject to two sources of instability: first, the approximation $\theta^\star(\phi) = \hat{\theta}(\phi)$ with a single gradient



(A) The 3 distributions over transformations forming the composition (B) Average of the 3 distrib.

FIGURE A.4: Evolution of the probability distributions π for CIFAR100 with unregularized unrolled optimization in a case of collapse.

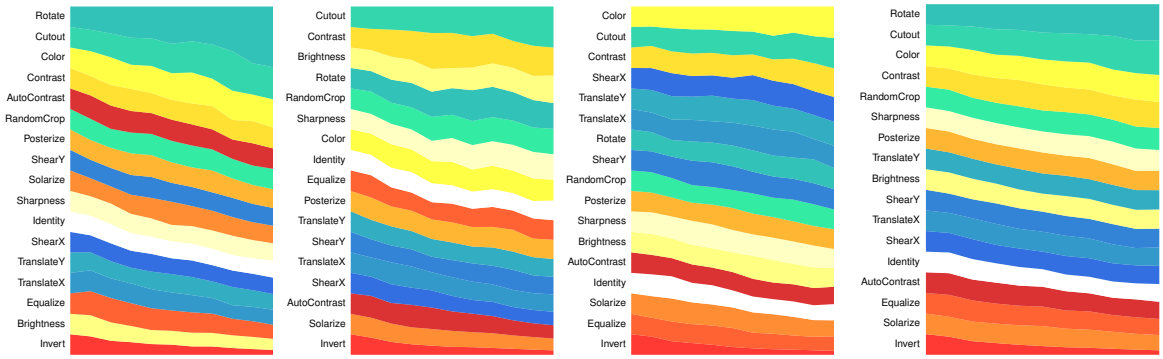


(A) The 3 distributions over transformations forming the composition (B) Average of the 3 distrib.

FIGURE A.5: Evolution of the distributions π for CIFAR100 with entropy-regularized unrolled optimization, on one of the search splits.

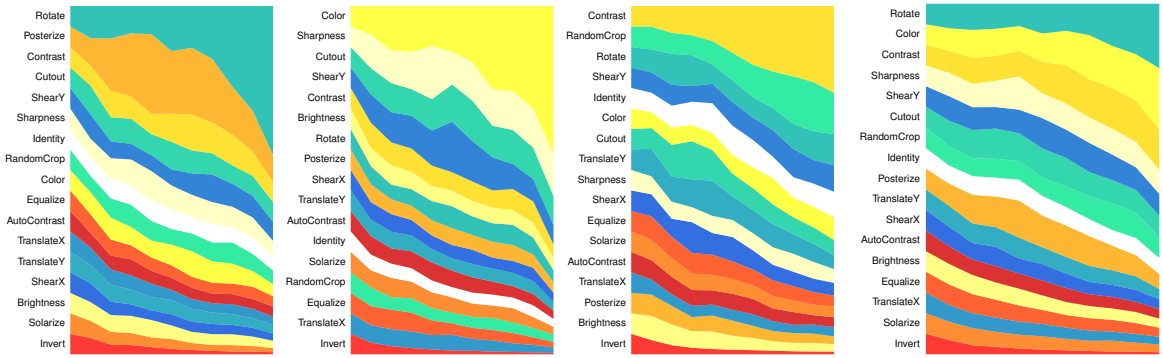
step inherently leads to wrong gradient updates; second, the REINFORCE gradient estimation is theoretically exact but has a high variance in practice when approximated in the context of stochastic optimization. Fig. A.4 illustrates these instabilities: blindly following wrong gradient directions exacerbated by an oversampling of the dominant transformation leads to a progressive collapse of the policy.

Unrolled optimization with entropy regularization. In the case of a single-stage unrolled optimization, the KL regularization uses a uniform distribution as an anchor, which corresponds to an entropy regularization. By maximizing the entropy, the algorithm encourages exploration of the augmentation policies and prevents the divergence phenomenon observed above. While this regularization leads to competitive results as reported in Table A.7, it does not mitigate the inherent instability of the gradient updates. On the other hand, the multi-stage algorithm we proposed in SLACK yields more stable gradient updates.



(A) 0.25: The three distributions over transformations forming the composition

(B) 0.25: Average of the 3 distributions



(C) 0.5: The three distributions over transformations forming the composition

(D) 0.5: Average of the 3 distributions

FIGURE A.6: Evolution of the probability distributions π for CIFAR100 with unregularized multi-stage optimization using upper-level learning rates of 0.25 and 0.5.

Multi-stage optimization without KL regularization. In our multi-stage approach, $\theta^\star(\tilde{\phi})$ is well-approximated at the beginning of each stage, as the model is re-trained with the current policy $\tilde{\phi}$. Gradient updates close to this policy are 'trusted' since our current θ after re-training stays close to $\theta^\star(\tilde{\phi})$, meaning that we strongly mitigate the approximation inherent to unrolled optimization. Our KL regularization encourages the policy to stay in this trust region and without it, the stochasticity of the optimization combined with the high variance from REINFORCE may drive the policy away. Fig. A.6 shows the evolution of our probability distributions under an unregularized multi-stage search with two different learning rates, one twice learger than the other. This evolution is smoother than with single-stage unrolled optimization and also quite stable when using a small learning rate, but this slows down convergence, yielding a sub-optimal policy (see Table A.7). The larger one leads again to a progressive divergence: the policy is driven too far and the $\theta^\star(\tilde{\phi})$ obtained after re-training becomes sub-optimal for the current ϕ. In other words, the KL regularization allows making large updates in the parameter space, while remaining close to a reference/anchor policy.

	CIFAR10		CIFAR100	
	WRN-40-2	WRN-28-10	WRN-40-2	WRN-28-10
SLACK	96.29 ± .08	97.46 ± .06	79.87 ± .11	84.08 ± .16
Ensembling of SLACK(4 GPUs)	96.33 ± .08	97.48 ± .06	79.94 ± .13	84.01 ± .14

TABLE A.8: CIFAR10/100 accuracy with ensembling strategy.

	CIFAR10		CIFAR100	
SLACKvariant	WRN-40-2	WRN-28-10	WRN-40-2	WRN-28-10
Warm-start	96.27 ± .09	97.05 ± .15	79.70 ± .11	83.90 ± .10
Cold-start (ours)	96.29 ± .08	97.46 ± .06	79.87 ± .11	84.08 ± .16

TABLE A.9: CIFAR10/100 accuracy with cold start and warm start.

A.3.4 An ensembling approach

In this section, we investigate the effect of an ensembling strategy for SLACK to reduce the variance of gradient updates in the search phase. More precisely, the strategy consists in independently training multiple models on the lower-level loss while averaging their contributions to the upper-level gradient. Each model is initialized (and subsequently re-initialized at each stage) based on a pre-training with a different seed. This ensembling strategy was implemented using multiple GPUs, where each GPU trains one copy of the model and only the upper-levels gradients are communicated and averaged across GPUs.

Results on CIFAR10/100 are reported in Table A.8. While there is a small improvement in most cases, the method still has a strong computational overhead. Yet it might be a relevant line of research for datasets for which the training procedure has a higher variance, e.g. smaller datasets where the additional cost of ensembling is not a significant overhead.

A.3.5 Warm-start vs cold-start

In this section, we study the model behaviour when searching with warm-start instead of cold-start. By warm-start, we mean that re-training is performed starting from the current network's weights at the beginning of each stage instead of re-initializing it to its pre-trained weights. We experimentally observe that warm-start with the same hyperparameters as for cold-start leads to a progressive overfitting of the network. Increasing the lower-level learning rate mitigates this phenomenon, but still yields sub-optimal results as reported in Table A.9. This suggests that re-training from $\theta_0^\star(\phi_0)$ gives a better estimate of $\theta^\star(\phi_i)$ at stage i than re-training from the biased state close to $\theta^\star(\phi_{i-1})$.

Appendix B

Appendix - On Good Practies for Task-Specific Distillation of Large Pretrained Visual Models

In this appendix, we first introduce additional experimental details regarding the choice of training hyperparameters and data augmentation, the prediction heads, PCA visualizations and training time (Appendix B.1). Next, we provide additional experimental results, with a comparison of linear and MLP heads for classification, an extended version of Tables 4.1 to 4.3 with confidence intervals, additional distillation results with EVA-02 MIM- and CLIP-pretrained models (Fang et al., 2023; Sun et al., 2023) instead of DINOv2, and a comparison of our chosen distillation loss (Hinton et al., 2015) to alternatives from the literature (Appendix B.2). Finally, we include additional visualizations of synthetic images produced by our mixing based on Stable Diffusion (Appendix B.3).

B.1 Additional experimental details

B.1.1 Datasets

Table B.1 reports the number of classes and the number of images in the training set of the datasets used for our study.

		Classes	Size (train)
DomainNet(Peng et al., 2019)	Painting	345	60617
	Sketch		56304
	Clipart		39064
Fine-grained classification	CUB (Wah et al., 2011)	200	5994
	Aircraft (Maji et al., 2013)	100	6667
	DTD (Cimpoi et al., 2014)	47	3760
Semantic segmentation	ADE20K (Zhou et al., 2017)	150	20210
	Cityscapes (Cordts et al., 2016)	19	2975
	Pascal VOC (aug.) (Everingham et al., 2010)	21	10582

TABLE B.1: Number of classes and size of training set of each dataset.

	Arch		Learning rate	Weight decay
DomainNet	ViT	Probing	$\{.0001, .0002, .0004\}$	0
		Finetuning/distillation	$\{1, 2, 4\} \times 10^{-5}$	$\{.025, .05, .1\}$
	ResNet-50	Training/distillation	.1	.0005
Fine-grained classification	ViT	Probing	$\{.001, .002, .004\}$	$\{.5, 1, 2, 4, 8\}$
		Finetuning/distillation	$\{1, 2, 4\} \times 10^{-5}$	$\{.025, .05, .1\}$
	ResNet-50	Training/distillation	$\{.01, .02, .04\}$	$\{.005, .01\}$
Semantic segmentation	ViT	Probing	.008	0
		Finetuning/distillation	$\{1, 2, 4\} \times 10^{-5}$	$\{.001, .01, .1\}$
	DeepLabv3(R50)	Training/distillation	$\{.01, .02, .04, .08\}$	$\{.0001, .001, .01\}$

TABLE B.2: Training hyperparameters for a batch size of 128 for DominNet and fine-grained classification tasks and 16 for segmentation tasks (32 for probing). Hyperparameters in {} are chosen based on a grid search on the validation set.

B.1.2 Training hyperparameters

Table B.2 details the weight decay and learning rate used for each task (classification on DomainNet, fine-grained classification, semantic segmentation), each architecture, and each training procedure (with/without freezing the pretrained backbone). Values are chosen based on a grid search on the validation set. More precisely, a coarse grid search is first performed on a logarithmic scale using powers of 10, before defining a finer one that is reported in Table B.2. When the grid search leads to values that are nearly identical for all tasks, we fix the value and report it in the table. Note that distillation with synthetic images based on Stable Diffusion is run with the best hyperparameters found for distillation without synthetic images. Also note that for finetuning experiments on ViT-L, we use a smaller batch size (8 for segmentation, 32 for classification) and reduce the learning rate in the grid search accordingly.

B.1.3 Generic data augmentation

In all experiments, we consistently apply classical data augmentation to both training and synthetic images (except for Mixup which is only applied to original images). The list of augmentations with their parameters is detailed below for each task (classification or segmentation) and architecture.

Classification. When training ViTs, we apply:

- RandomResizedCrop with scale 0.08
- ColorJitter with range $(0, 0.4)$
- RandomFlip with probability 0.5
- Mixup with parameter 0.2.

An exception is for probing on fine-grained classification tasks, where we simply apply Resize and CenterCrop instead of RandomResizedCrop, and do not apply Mixup. We found that these transformations were too strong for fine-grained classification with a frozen backbone.

When training the ResNet-50, we use TrivialAugment's (Müller and Hutter, 2021) strategy (ImageNet version), that consists of

- RandomResizedCrop with scale 0.08
- ColorJitter with range $(0, 0.4)$
- RandomFlip with probability 0.5
- A fourth transformation randomly sampled among a pool.

However for fine-grained classification, we use a scale parameter of 0.4 for RandomResizedCrop, as 0.08 proved too strong for training from scratch on fine-grained tasks.

For validation and testing, we follow the standard procedure of applying Resize and CenterCrop.

Semantic segmentation. We train with images of size (s, s) with $s = 560$. We apply the following data augmentations for all experiments, which correspond to the mmsegmentation augmentations also used by DINOv2:

- Resize to $(., s)$ with ratio range $(0.5, 2.0)$
- RandomCrop to (s, s), with cat_max_ratio $= 0.75$
- RandomFlip with probability 0.5
- PhotoMetricDistortion.

For validation and testing, we use sliding windows of size (s, s) and stride $\frac{s}{2}$.

B.1.4 Details on prediction heads

We remind the reader that for segmentation, we use the same linear evaluation head as DINOv2's (Oquab et al., 2024) while for classification, we use a MLP head, unlike DINOv2 which uses a linear head.

More precisely, let $n_{\text{in}}, n_{\text{hidden}}, n_{\text{out}}$ respectively denote the number of input, hidden, and output neurons in the MLP head. n_{out} is the number of classes, and n_{in} the number of input features extracted from the pretrained backbone, meaning that $n_{\text{in}} = n_f \times (n_{\text{CLS}} + \mathbf{1}_{\textbf{use avgpool}})$, where

- n_f is the embedding dimension ($n_f = 1536, 1024, 384$ for ViT-g, ViT-L and ViT-S respectively)
- n_{CLS} is the number of blocks from which the CLS tokens are concatenated ($n_{\text{CLS}} = 4$ for DomainNet, 3 for fine-grained tasks)
- $\mathbf{1}_{\textbf{use avgpool}}$ indicates whether we also concatenate the average pooling of the patch embeddings of the last block (true for DomainNet).

As for the number of hidden neurons n_{hidden}, we set $n_{\text{hidden}} = n_{\text{in}}$ for ViT-S and $n_{\text{hidden}} = \sqrt{n_{\text{in}} \times n_{\text{out}}}$ for ViT-L/ViT-g, as we experimentally found that this choice gave the best results. Intuitively, using such intermediate size for ViT-L/ViT-g, whose embedding sizes are larger (1024 and 1536), allow for a more progressive decrease toward n_{out}.

B.1.5 Details on the PCA

The main paper provides PCA-based visualizations of the learned representations for CUB and ADE20K datasets, respectively in Fig. 4.1 and 4.4. Detailed step-by-step descriptions of how these visualizations were constructed are provided below.

For the PCA visualization of teacher predictions from Fig. 1 on the CUB fine-grained classification task, the steps are the following:

1. **Feature computation** for both original and synthetic CUB training images, giving class token predictions of shapes (N, D) and $(N_{\text{synthetic}}, D)$ with D the embedding dimension ($D = 1536$ for ViT-g and 384 for ViT-S)
2. **Subsampling**: we only keep the first 20 classes. We keep synthetic images that result from a mix of images belonging to this set of 20 classes. This leaves $M < N$ and $M_{\text{synthetic}} < N_{\text{synthetic}}$ images.
3. **PCA** over the (M, D) predictions on original images
4. **Visualization** of the $(M + M_{\text{synthetic}}, D)$ data points projected onto the two main principal components, colored by class label for the M images, and in gray for the $M_{\text{synthetic}}$ images.

For the visualization of student predictions on Figure 1, the steps are the same but without the synthetic images.

For the PCA visualization from Figure 4 on the ADE20K segmentation task, we visualize patch embedding representations as follows:

1. **Feature computation** on $N = 500$ test images, giving patch embedding predictions of shape (N, D, H, W) with D the embedding dimension; and $H = W = 40$ $(= \frac{\text{image size}}{\text{patch size}} = \frac{560}{14})$.
2. **Resizing** of the corresponding 500 segmentation maps to shape (N, C, H, W) where C is the number of classes. We use mmsegmentation's resize method on one-hot encoded labels.
3. **Flattening** of predictions and labels to $(N \times H \times W, D)$, $(N \times H \times W, C)$ respectively.
4. **Filtering**: we keep patches whose labels are well defined, with a probability over 0.9.
5. **Subsampling**: we only keep 20 classes. We select those whose size (number of patches of this class) is closest to the median size.
6. **Filtering** and **subsampling** yield a number M of data points, $M < N \times H \times W$.
7. **PCA** on the (M, D) predictions.
8. **Visualization** of the two main principal components, colored by class label.

B.1.6 Training time

All our experiments were performed on a single GPU (either V100 or A100). We detail the training time with a pretrained ViT-g as teacher and a pretrained ViT-S as student. When using a ResNet-50 from scratch as student, the training time per epoch is similar to that of the ViT-S, but we train for longer (200 epochs instead of 80). Experimentally, we observed that probing the teacher (ViT-g) either takes less time or about the same amount of time as finetuning the student (ViT-S). Distillation with the probed ViT-g as teacher takes approximately twice as long as finetuning. Lastly, adding data augmentation based on Stable Diffusion further increases the training time by 1.5 times in average. For example, finetuning the ViT-S for ADE20K takes 16 hours on a A100 GPU, while distillation with data augmentation based on Stable Diffusion takes 55 hours, and probing the ViT-g takes 14 hours.

As for the generation of synthetic data, our image mixing procedure (Pinkney, 2022) roughly takes 2 hours for 1000 images (on a V100 GPU). We remind that we generated synthetic datasets with n times more images than in the training set, with $n = 5$ for DomainNet and semantic segmentation, and $n = 10$ for the relatively smaller fine-grained tasks. The size of the training set of each task is reported in Table B.1.

Model		Head	Classification on DomainNet			Fine-grained classification			Semantic segmentation		
			Painting	Sketch	Clipart	CUB	Aircraft	DTD	ADE20K	Cityscapes	VOC
ViT-g	Probing	Linear - Oquab et al. (2024)	-	-	-	91.6	87.2	84.5	49.0	71.3	83.0
		Linear - ours	82.3	80.5	84.9	91.7	87.8	85.5	48.8	71.2	83.5
		MLP - outs	83.0	81.2	85.7	91.6	88.1	85.8	-	-	-
ViT-L	Probing	Linear - Oquab et al. (2024)	-	-	-	90.5	81.5	84.0	47.7	70.3	82.1
		Linear - ours	82.2	79.9	85.1	91.8	86.5	85.4	47.8	70.4	82.7
		MLP - ours	82.9	80.4	85.3	91.3	87.8	85.5	-	-	-
ViT-S	Probing	Linear - Oquab et al. (2024)	-	-	-	88.1	74.0	80.6	44.3	66.6	81.1
		Linear - ours	75.6	70.0	78.7	89.0	75.8	80.8	45.1	67.0	81.8
		MLP - ours	77.3	71.9	79.3	88.2	77.1	82.1	-	-	-
	Finetuning	Linear - ours	78.5	75.6	81.3	87.0	87.3	81.2	49.8	75.8	84.6
		MLP - ours	79.4	76.0	81.8	87.3	87.8	81.6	-	-	-

TABLE B.3: **Comparison of our linear probing results with DINOv2's (Oquab et al., 2024) and comparison of linear and MLP heads for classification.** We report accuracy for classification and mIoU for segmentation. We report results from probing ViT-g, ViT-L, ViT-S, and from finetuning ViT-S. The underlined figures correspond to those in Table 4.1.

Model		Classification on DomainNet (acc)			Fine-grained classification (acc)			Semantic segmentation (mIoU)		
		Painting	Sketch	Clipart	CUB	Aircraft	DTD	ADE20K	Cityscapes	VOC
ViT-S	(1a) Probing	77.3 ±.2	71.9 ±.3	79.3 ±.2	88.2 ±.1	77.1 ±.4	82.1 ±.5	45.1 ±.1	67.0 ±.2	81.8 ±.2
	(1b) Finetuning	79.4 ±.3	76.0 ±.2	81.8 ±.2	87.3 ±.8	87.8 ±.9	81.6 ±1.1	49.8 ±.4	75.8 ±.3	84.6 ±.7
ViT-L	(2a) Probing	82.9 ±.2	80.4 ±.2	85.3 ±.1	91.3 ±.5	87.8 ±1.3	85.5 ±.3	47.8 ±.2	70.4 ±.1	82.7 ±.3
	(2b) Finetuning	83.9 ±.2	81.4 ±.1	85.9 ±.1	91.5 ±.1	94.0 ±.5	85.8 ±.6	57.4 ±.1	78.6 ±.2	88.0 ±.4
ViT-g	(3a) Probing	83.0 ±.4	81.2 ±.2	85.7 ±.1	91.6 ±.2	88.1 ±.6	85.8 ±.3	48.8 ±.2	71.2 ±.2	83.5 ±.1

TABLE B.4: **Probing/finetuning of DINOv2 pretrained models** for classification on DomainNet, fine-grained classification and semantic segmentation. We report accuracy for classification and mIoU for segmentation. We report the 95% confidence estimation (1.96σ) averaged to the upper decimal.

B.2 Additional experimental results

B.2.1 Additional results with a linear prediction head

In Table B.3, we compare classification results using linear and MLP heads when probing ViT-S, ViT-L and ViT-g, and when finetuning ViT-S. We also compare our linear evaluation results with DINOv2's (Oquab et al., 2024). For linear probing, our performance is similar to DINOv2's. Relative differences can be attributed to varying choices of data augmentation for classification, and to differences in image size for segmentation (we train with size 560 while DINOv2 uses 512). For classification, we observe that using a MLP head consistently improves results, both for probing and finetuning. In other words, using a MLP head for both the teacher and student models independently boosts their accuracy, and we experimentally observed that it also makes a better teacher, as it yields the best distillation results for the student.

Student	Teacher		SD	Classification on DomainNet (acc)			Fine-grained classification (acc)			Semantic segmentation (mIoU)		
				Painting	Sketch	Clipart	CUB	Aircraft	DTD	ADE20K	Cityscapes	VOC
ViT-S	ViT-S probed	(4a)	✗	80.0 ±.2	76.9 ±.4	82.2 ±.3	89.4 ±.3	86.5 ±.7	82.9 ±.4	49.6 ±.2	71.2 ±.3	84.6 ±.2
		(4b)	✓	80.2 ±.3	77.1 ±.2	82.4 ±.5	89.7 ±.3	86.6 ±.6	83.4 ±.5	50.3 ±.2	72.3 ±.3	84.9 ±.2
	ViT-L probed	(5a)	✗	80.5 ±.3	77.8 ±.3	**83.4** ±.2	89.7 ±.4	89.2 ±.7	83.4 ±.4	50.7 ±.3	74.0 ±.2	85.5 ±.3
		(5b)	✓	**80.8** ±.3	**78.0** ±.2	83.2 ±.4	**90.0** ±.3	**89.8** ±.4	**84.0** ±.4	**51.7** ±.2	74.7 ±.2	**86.1** ±.3
	ViT-L finetuned	(6a)	✗	79.7 ±.3	77.0 ±.2	82.5 ±.3	88.6 ±.4	88.9 ±.6	81.5 ±.7	50.7 ±.5	**76.3** ±.3	84.8 ±.4
		(6b)	✓	80.3 ±.2	77.2 ±.2	82.9 ±.3	88.6 ±.3	89.1 ±.4	82.5 ±.5	51.6 ±.7	**76.4** ±.3	85.7 ±.4
	ViT-g probed	(7a)	✗	80.5 ±.1	77.7 ±.3	**83.4** ±.2	89.1 ±.5	**89.6** ±.5	83.1 ±.8	51.6 ±.4	74.4 ±.2	85.7 ±.3
		(7b)	✓	**80.8** ±.2	**78.0** ±.2	**83.3** ±.3	**89.8** ±.4	**90.1** ±.7	**83.6** ±.6	**52.1** ±.4	75.0 ±.1	**86.3** ±.2

TABLE B.5: **Distillation on ViT-S initialized with DINOv2** for classification on DomainNet, fine-grained classification and semantic segmentation. We report accuracy for classification and mIoU for segmentation. We report results with and without data augmentation based on Stable Diffusion (SD), for various choices of teachers. We report the 95% confidence estimation (1.96σ) averaged to the upper decimal. **Bold** numbers: within 95% confidence interval of the best score for each task.

Student	Teacher		SD	Classification on DomainNet (acc)			Fine-grained classification (acc)			Semantic segmentation (mIoU)		
				Painting	Sketch	Clipart	CUB	Aircraft	DTD	ADE20K	Cityscapes	VOC
R50	-	(8a)	-	66.0 ±.4	68.1 ±.3	72.5 ±.9	73.3 ±.2	85.0 ±.9	63.5 ±1.2	37.8 ±.7	**67.9** ±.7	67.5 ±1.0
	ViT-g probed	(9a)	✗	67.7 ±.7	70.5 ±1.0	**74.9** ±.4	76.0 ±.6	85.7 ±.4	66.7 ±.6	38.2 ±.6	**67.7** ±1.6	67.7 ±.5
		(9b)	✓	**69.1** ±.8	**71.0** ±.3	**75.2** ±.6	**79.1** ±.9	**87.8** ±.5	**69.4** ±1.1	**42.1** ±.4	**69.3** ±1.9	**73.9** ±.7

TABLE B.6: **Distillation from ViT-g to ResNet-50 (resp. DeepLabv3-ResNet50 for segmentation) trained from scratch** for classification on DomainNet, fine-grained classification and semantic segmentation. We report accuracy for classification and mIoU for segmentation. We report results with and without data augmentation based on Stable Diffusion (SD). We report the 95% confidence estimation (1.96σ) averaged to the upper decimal. **Bold** numbers: within 95% confidence interval of the best score for each task.

B.2.2 Main results with standard deviations

In Tables B.4 to B.6, we provide uncertainty estimations to the results reported in Tables 4.1 to 4.3 of the main paper, respectively. We use 1.96σ as 95% confidence estimator, and report its value averaged to the upper decimal, where σ is the standard deviation of the results obtained through different runs. We remind that we use 3 runs for probing, finetuning and training, $3 \times 2 = 6$ runs for distillation on DomainNet and $3 \times 3 = 9$ runs for distillation on fine-grained and semantic segmentation tasks.

B.2.3 Distillation with EVA-02 pretrained models

In this section, we assess whether the good practices we have drawn as experimental conclusions of our study using DINOv2's pretrained models as teachers also transfer to EVA-02 (Fang et al., 2023) and EVA-02-CLIP (Sun et al., 2023) models, that were pretrained with masked image modeling (MIM) and CLIP training, respectively. More precisely, we use i) either EVA-02 or EVA-02-CLIP ViT-L models as teachers, pretrained on datasets composed of 38 million images and 2 billion images respectively, and ii) EVA-02 ViT-S model as a student, pretrained on ImageNet-21k.

Model		Classification on DomainNet			Fine-grained classification		
		Painting	Sketch	Clipart	CUB	Aircraft	DTD
EVA-02 ViT-S	(1a) Probing	51.9	31.7	56.7	45.4	31.4	53.7
	(1b) Finetuning	80.1	76.0	82.3	88.3	84.8	81.1
EVA-02 ViT-L	(2a) Probing	84.1	82.4	87.0	85.1	63.5	83.9
	(2b) Finetuning	85.2	83.2	86.7	91.8	90.7	86.8
EVA-02-CLIP ViT-L	(2a) Probing	83.9	82.4	86.9	85.2	63.9	83.4
	(2b) Finetuning	85.3	83.4	86.6	91.5	93.5	86.9

TABLE B.7: **Probind/finetuning of EVA-02 (Fang et al., 2023) and EVA-02-CLIP (Sun et al., 2023) pretrained models** for classification on DomainNet and fine-grained classification. We report the probing/finetuning accuracies of EVA-02 ViT-L, EVA-02-CLIP ViT-L used as teachers, and EVA-02 ViT-S used as student. Relative distillation gains in Table B.8 are with respect to underlined results in this table.

Student	Teacher		SD	Classification on DomainNet			Fine-grained classification		
				Painting	Sketch	Clipart	CUB	Aircraft	DTD
EVA-02 ViT-S	EVA-02 ViT-L probed	(5a)	✗	81.0 (+0.9)	**78.0** (+2.0)	**83.7** (+1.4)	87.3 (-1.0)	78.9 (-5.9)	**83.6** (+2.5)
		(5b)	✓	**81.4** (+1.3)	**78.1** (+2.1)	**83.8** (+1.5)	87.6 (-0.7)	81.3 (-3.5)	**83.9** (+2.8)
	EVA-02 ViT-L finetuned	(6a)	✗	80.2 (+0.1)	76.6 (+0.6)	82.5 (+0.2)	**88.4** (+0.1)	**85.6** (+0.8)	81.6 (+0.5)
		(6b)	✓	80.5 (+0.4)	77.2 (+1.2)	83.1 (+0.8)	**88.6** (+0.3)	**86.2** (+1.4)	82.8 (+1.7)
	EVA-02-CLIP ViT-L probed	(5a)	✗	81.0 (+0.9)	**78.0** (+2.0)	**83.9** (+1.6)	87.4 (-0.9)	79.0 (-5.8)	82.9 (+1.8)
		(5b)	✓	81.2 (+1.1)	**78.2** (+2.2)	**83.7** (+1.4)	87.4 (-0.9)	81.7 (-3.1)	**83.5** (+2.4)
	EVA-02-CLIP ViT-L finetuned	(6a)	✗	79.9 (-0.2)	76.8 (+0.8)	82.6 (+0.3)	**88.5** (+0.2)	**85.7** (+0.9)	81.3 (+0.2)
		(6b)	✓	80.5 (+0.4)	77.0 (+1.0)	82.9 (+0.6)	**88.8** (+0.5)	**86.4** (+1.6)	82.7 (+1.6)

TABLE B.8: **Distillation with EVA-02 (Fang et al., 2023) and EVA-02-CLIP (Sun et al., 2023) pretrained models** for classification on DomainNet and fine-grained classification. We report results with and without data augmentation based on Stable Diffusion (SD), using EVA-02 ViT-S as student and EVA-02 ViT-L, EVA-02-CLIP ViT-L as teachers. Relative gains w.r.t. to underlined result from Table B.7 are in parentheses. **Bold** numbers: within 95% confidence interval of the best score for each task.

Results are reported in Tables B.7 and B.8. The line numbering is kept consistent with that of DINOv2 experiments. Compared to DINOv2, probing results for ViT-L are stronger on DomainNet but weaker on fine-grained tasks, especially on Aircraft (2a). Probing results for ViT-S are overall very low compared to DINOv2's ViT-S (1a), which is certainly due to the fact that EVA-02's ViT-S was pretrained from scratch while DINOv2's ViT-S was distilled from their ViT-g. Distillation with a probed ViT-L is detrimental for CUB and Aircraft (5). On the four other tasks, it boosts results and outperforms distillation from a finetuned ViT-L (5 vs 6). Similarly to the case of Cityscapes with DINOv2 models (Tables 4.1 and 4.2), poor distillation results with the probed ViT-L for CUB and Aircraft can be explained by fact that the finetuned ViT-S student outperforms the probed ViT-L teacher by a large margin (resp. 3.2% and 21.3% accuracy).

	$\mathcal{L}_{\text{distill}}$	Mixup	CUB	Aircraft	DTD
Finetuning	-	✓	87.3	87.8	81.6
Distillation	Hard-label KD (Touvron et al., 2021)	✗	88.1	88.3	81.2
	CRD (Tian et al., 2020)	✓	89.0	88.4	83.3
	DKD (Zhao et al., 2022)	✗	88.5	85.7	82.8
	KD (Hinton et al., 2015)	✗	88.6	89.0	82.6
		✓	89.1	89.6	83.1

TABLE B.9: **Choice of distillation loss $\mathcal{L}_{\text{distill}}$.** Comparison of the classical distillation loss KD introduced by Hinton et al. (2015), chosen for our study, with i) hard-label distillation, used by Touvron et al. (2021), ii) CRD (Tian et al., 2020), and iii) DKD (Zhao et al., 2022). We evaluate the version of CRD that uses KD and a temperature $\tau = 0.5$, as it gave the best results. Note that the Mixup augmentation is not applied for hard-label distillation and DKD, as these losses are not compatible with Mixup. Results are with ViT-g as teacher and ViT-S as student, withhout data augmentation based on Stable Diffusion.

B.2.4 Ablation for the distillation loss

We compare the distillation loss $\mathcal{L}_{\text{distill}}$ originally proposed by Hinton et al. (2015) (and that we used in all experiments in the main paper) to other more recent alternatives from the literature: i) Hard-label distillation, that consists in a cross-entropy loss with the hard label prediction produced by the teacher and is employed by Touvron et al. (2021) for learning their distillation token (here, to be agnostic to the architecture, we simply reuse their loss, not the full distillation protocol), ii) CRD (Tian et al., 2020), that aligns teacher and student representations through a contrastive learning objective, and iii) DKD (Zhao et al., 2022), that decomposes the classical knowledge distillation loss into target-class and non-target-class losses.

Results for fine-grained classification are reported in Table B.9 and show that, for the task-specific distillation of DINOv2 pretrained models, the classical knowledge distillation loss introduced by Hinton et al. (2015) performs best.

B.3 Additional visualizations

Visualizations of the synthetic images produced by our augmentation protocol based on stable-diffusion for ADE2OK (Zhou et al., 2017), DomainNet's Sketch (Peng et al., 2019) and DTD (Cimpoi et al., 2014) can be found in Fig. B.1.

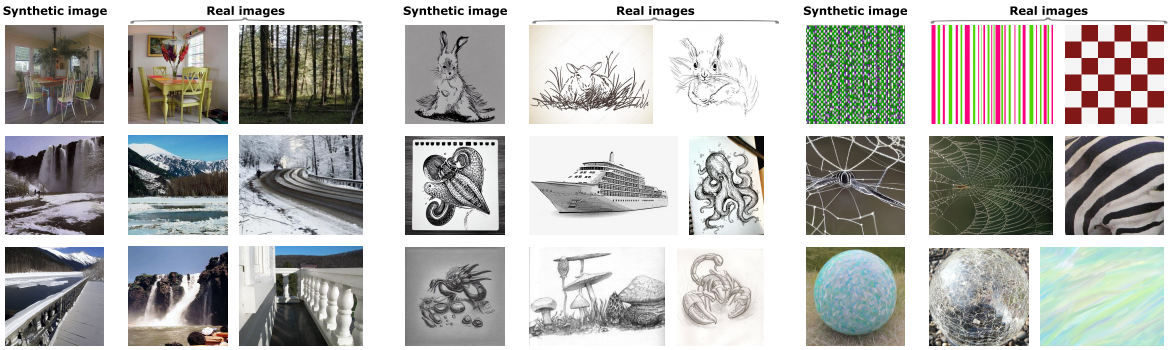


FIGURE B.1: **Diffusion-based data augmentation**. Examples of synthetic images generated using ImageMixer (Pinkney, 2022) as described in Sec. 4.3.2, mixing two training images from ADE20K (Zhou et al., 2017) (left), Sketch from DomainNet (Peng et al., 2019) (middle) and DTD (Cimpoi et al., 2014) (right). Those populate the extended dataset $\mathcal{D}_{\text{sd}}$ used for distillation.

Appendix C

Appendix - LUDVIG: Learning-free Uplifting of 2D Visual features to Gaussian Splatting scenes

C.1 Using LUDVIG for downstream tasks

In this section, we describe our approach for uplifting DINOv2, SAM and CLIP models and evaluating the 3D features on two downstream tasks: segmentation and open-vocabulary object detection.

As in Sec. 5.3.2, we are given a set of 2D frames $I_1, \ldots, I_m$, with viewing directions $d_1, \ldots, d_m$ and a corresponding 3D scene obtained using the Gaussian Splatting method.

Multi-view segmentation. For this task, we assume that a foreground mask of the object to be segmented is provided on a reference frame taken to be the first frame I_1. We consider two types of foreground masks: either scribbles or a whole reference mask of the object, both of which define a set of foreground pixels $\mathcal{P}$. In the following subsections, we present the proposed approaches for segmentation using SAM and DINOv2 features, based on both types of foreground masks.

C.1.1 Multi-view segmentation with SAM

SAM (Kirillov et al., 2023; Ravi et al., 2024) is a powerful image segmentation model, that can generate object segmentation masks from point prompts on a single 2D image. Aggregating SAM 2D segmentation masks in 3D allows for cross-view consistency and improves single-view segmentation results. In order to leverage SAM, we propose a simple mechanism for generating SAM 2D features for each frame from a foreground mask in the reference frame.

Generating 2D query points for SAM. The key idea is to generate point prompts on each frame from the foreground mask provided on the reference frame. To this end,

we perform an uplifting of the foreground mask (Eq. (5.5) from the main paper) and re-project it on all frames (Eq. (5.4) from the main paper). From the reprojected mask for viewing direction d, further normalized by its average value, we retain a subset $\mathcal{P}_d$ of pixels with values higher than a threshold τ fixed for all scenes ($\tau = 0.4$ for SPIn-NeRF and $\tau = 1$ for NVOS). We then predict a SAM mask based on these point prompts as described below.

Predicting a SAM segmentation mask from a set of query points. Given a set of pixels $\mathcal{P}_d$ pertaining to the foreground, we compute 2D segmentation masks using SAM by randomly selecting 3 points prompts from $\mathcal{P}_d$, repeating the operation 10 times and averaging the resulting masks for each view to obtain the final 2D SAM feature maps.

Segmentation with uplifted SAM masks. The 2D segmentation masks generated by SAM are uplifted using the aggregation scheme described in Sec. 5.3. Our final prediction is obtained by rendering the uplifted feature maps into the target frame and thresholding.

Evaluation and hyperparameter tuning. Segmentation with 3D SAM masks requires setting a threshold for foreground/background pixel assignment, and optionally choosing one of the three masks proposed by SAM (representing different possible segmentations of the object of interest). For SPIn-NeRF, the threshold and mask prediction are chosen based on the IoU for the available reference mask. For NVOS, only reference scribbles are provided; hence, a single mask is predicted, and the segmentation threshold is fixed across all scenes for SAM, and automatically chosen using Li's iterative Minimum Cross Entropy method (Li and Lee, 1993) for SAM 2.

C.1.2 Multi-view segmentation with DINOv2

DINOv2 (Oquab et al., 2024) is a self-supervised vision model recognized for its generalization capabilities. In this work, we aggregate the patch-level representations produced by DINOv2 with registers (Darcet et al., 2024) into a high resolution and fine-grained 3D semantic representation.

Construction of 2D feature maps. We construct the 2D feature maps using a combination of a sliding windows mechanism and dimensionality reduction of the original DINOv2 features. Specifically, we i) extract DINOv2 patch-level representations across multiple overlapping crops of each image, ii) apply dimensionality reduction over the set of all patch embeddings, ii) upsample and aggregate the dimensionality-reduced

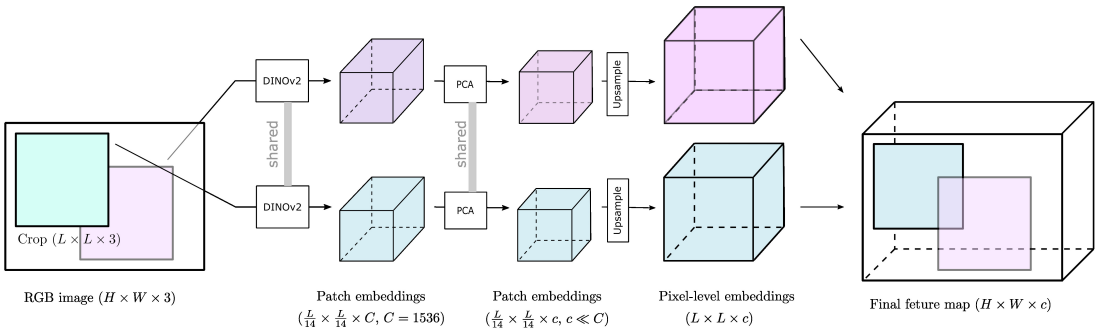


FIGURE C.1: Sliding windows for construction of DINOv2 feature maps.

patch embeddings to obtain pixel-level features for each image. The process is illustrated in Fig. C.1. This approach enhances the granularity of spatial representations by aggregating patch-level representations to form pixel-level features. To favor the first principal components, known to focus on the foreground objects (Oquab et al., 2024), the features are re-weighted by the eigenvalues of the PCA decomposition.

Segmentation with uplifted DINOv2 features. The 2D feature maps from the m views are uplifted using Eq. (5.5) from the main paper, and the resulting 3D features are re-projected into any viewing direction d using Eq. (5.4) from the main paper to compute rendered 2D features $(\hat{F}_{d,p})$. To obtain segmentation masks, we construct a score $P(\hat{F}_{d,p})$ for a 2D pixel p to belong to the foreground, based on its corresponding rendered feature. More precisely, P relies on the rendered foreground features $\mathcal{F}_{\text{ref}} := (\hat{F}_{d_1,p})_{p\in\mathcal{P}}$ corresponding to the foreground mask computed on the reference frame I_1. We propose two approaches for constructing P. The first one is a simple approach that sets $P(\hat{F}_{d,p}) = \mathcal{S}_F(\hat{F}_{d,p}, \bar{F})$ where $\bar{F}$ is the average over foreground features $\mathcal{F}_{\text{ref}}$, and $\mathcal{S}_F$ is defined based on the cosine similarity. The second approach is more discriminative and first trains a logistic regression model P on all rendered 2D features of the reference frame, so that the foreground features $\mathcal{F}_{\text{ref}}$ are assigned a positive label. Then $P(\hat{F}_{d,p})$ gives the probability that a pixel p belongs to the foreground. The final mask is then obtained by thresholding.

Experimentally, the second approach is extremely efficient when the set of foreground pixels $\mathcal{P}$ covers the whole object to segment, so that P captures all relevant features. This is the case when a whole reference mask of the object is provided. When the foreground pixels $\mathcal{P}$ does not cover the whole object, as with scribbles, P can be discriminative to parts of the object that are not covered by $\mathcal{P}$. Therefore, we rely on the second approach for tasks where a reference mask is provided, and use the simpler first approach when only scribbles serve as reference.

C.1.3 Improving DINOv2 segmentation with graph diffusion

DINOv2 provides generic visual features that do not explicitly include information for segmentation, unlike models such as SAM that were specifically trained for such a task. Consequently, using the 2D projections of uplifted DINOv2 features, as proposed in Sec. C.1.2, might fail to separate different objects that happen to have similar features while still being distinct entities. This challenge can be mitigated by incorporating 3D spatial information in which the objects are more likely to be well-separated. To this end, we propose to leverage the graph diffusion process introduced in Sec. 5.3.3. Below, we describe the initialization of the weight vector g_0 and the construction of the adjacency matrix A.

Initialization of the weight vector. The initial vector of weights $g_0 \in \mathbb{R}^n$ representing a coarse estimation of the contribution of each Gaussian to the segmentation mask. It is computed by uplifting the 2D foreground mask (either scribbles or a reference mask) from the reference frame using Eq. (5.3) from the main paper, and retaining only the top 10% of Gaussians with positive mask values, setting the rest to zero. The nodes for which g_0 has a positive value define a set of anchor nodes $\mathcal{M}$ that are more likely to contribute to the foreground. The resulting weight vector is a coarse estimation of how much each Gaussian contributes to a rendered 2D segmentation mask.

Construction of graph edges. We define the pairwise similarity function S_f as:

$$S_f(f_i, f_j) = \exp\left(-\frac{||f_i - f_j||_2^2}{bs_f^2}\right) \tag{C.1}$$

where f_i, f_j are the l_2-normalized DINOv2 features, s_f is the median of pairwise l_2 distances and b is a bandwidth parameter. We choose a global unary regularization term $P(f_i)$ on each node i to contain diffusion to nodes with features similar to those of the foreground. More precisely, P is defined using a similar approach as in Sec. C.1.2:

1. When scribblers are provided, $P(f_i) = \mathcal{S}_f(f_i, \bar{f})$ with the averaged feature $\bar{f}$ over the anchor nodes $\mathcal{M}$, and a different value for the bandwidth b.
2. When a full foreground mask is available, we train a logistic regression model on the uplifted features with positive labels for the anchor nodes' features. The unary term is then defined as $P(f_i) = \mathcal{L}(f_i)^{1/b}$, with b a bandwidth parameter and $\mathcal{L}(f_i)$ is the model's predicted probability for f_i.

The local term S_f allows diffusing to neighbors that have similar features while the unary term prevents leakage to background nodes and allows using an arbitrary number of diffusion steps.

The matrix A of graph edges is then defined based on S_f and P as in Eq. (5.7) from the main paper. For this task, we binarize A with a fixed threshold (set to 10^{-5}). After the T diffusion steps, we recover the nodes $\mathcal{S}$ in g_T with strictly positive values (i.e., those reachable after T iterations). The final weight is defined as $h_i = P(f_i)$ if $i \in \mathcal{S}$ and 0 otherwise. Segmentation is then performed by projecting $\mathbf{h} = (h_i)$ into 2D and thresholding. The selection process of the threshold and bandwidth parameters is detailed below.

Evaluation and hyperparameter tuning. Segmentation relies on three hyperparameters: the bandwidths for S_f and P, and the threshold for foreground/background pixel assignment. For SPIn-NeRF, all hyperparameters are chosen based on the IoU for the available reference mask. For NVOS, only reference scribbles are provided, hence we predict a SAM mask based on the scribbles of the reference frame, and choose the hyperparameters maximizing the IoU with this SAM mask. This is consistent with a scenario where the user, here SAM, would choose hyperparameters based on visual inspection on one of the frames.

C.1.4 Open-vocabulary object localization

For the open-vocabulary object localization task, multi-resolution CLIP feature maps are constructed as described in Sec. 5.4.2 and uplifted along with DINOv2 features using Eq. (5.3) from the main paper. The refined 3D CLIP features are then evaluated on the LERF localization and segmentation tasks as described below.

C.1.4.1 Relevancy scores and object localization.

We consider uplifted 3D CLIP features f. We follow LERF (Kerr et al., 2023) and LangSplat (Qin et al., 2024) to compute alignment scores between CLIP visual features and a text query, denoted as relevancy score, and for object localization based on these relevancy scores.

Relevancy scores. The relevancy $r_{i,q}$ between a feature f_i and text query ϕ_q is defined as follows:

$$r_{i,q} = \min_k \frac{\exp(T \cdot f_i \cdot \phi_q)}{\exp(T \cdot f_i \cdot \phi_q) + \exp(T \cdot f_i \cdot \phi^k_{cano})}, \tag{C.2}$$

where T is a temperature parameter set to 10 by Kerr et al. (2023) and ϕ^k_cano is the CLIP embedding of a predefined canonical phrase chosen from "object," "things," "stuff," and "texture." Note that Qin et al. (2024) compute the relevancy scores for 2D pixels, while we directly compute them for 3D Gaussians, allowing their manipulation in 3D.

Localization. The 3D CLIP relevancies can be projected into 2D for a given camera pose, and used for localization and segmentation for each text query. For localization, we follow Qin et al. (2024) and choose the pixel with the highest relevancy score, following a 2D smoothing using a mean filter with kernel size $K = 5$ in our work.

C.1.4.2 Object segmentation

Segmentation based on raw CLIP relevancies is challenging, as fully covering the object of interest without capturing other objects of a similar nature is challenging.

We first describe our process for segmenting directly based 3D relevancies. We then present two complementary approaches that allow for a more targeted segmentation: predicting 2D SAM masks by retrieving query points with high relevancies, and refining 3D relevancy scores using graph diffusion based on 3D DINOv2 features.

Segmentation from 3D relevancies. Given a camera pose and a text query, a segmentation mask is obtained by first applying a rough thresholding over projected relevancies rescaled by their maximum value, with a fixed threshold value $\tau = 0.8$, followed by automatic thresholding with Otsu's method (Otsu et al., 1975).

Improving 2D segmentation with SAM. For segmentation with SAM, we use the pixels with the highest relevancy scores as query points for a given camera pose and text query. More precisely, we first obtain a mask $\mathcal{M}$ by projecting and thresholding the relevancies as described above, using $\tau = 0.93$. We then use the approach described in Sec. C.1.1, paragraph Predicting a SAM segmentation mask from a set of query points and average 20 mask predictions. We choose the top-q percent of pixels as the subset of query points for SAM, where q is the proportion of positive pixels in $\mathcal{M}$, hence extracting a larger subset of point prompts for larger objects. The scalar map obtained by averaging the 20 predicted masks is then automatically thresholded again using Otsu's method (Otsu et al., 1975).

Refining 3D CLIP relevancies with graph diffusion based on DINOv2 features. We refine 3D CLIP relevancy scores using graph diffusion based on 3D DINOv2 features (f), as in Sec. C.1.3. The diffusion process runs in parallel for all text queries. For initialization, we keep positive a very restricted set of nodes with the highest relevancy, whose weight propagate to neighboring nodes with similar DINOv2 features, progressively spanning the object of interest. The diffusion process results in a better segmentation both with and without leveraging SAM. When using SAM, the set of query pixels can have a larger coverage of the object of interest without extending to other objects.

Details on graph construction and node initialization for refining 3D CLIP relevancies. The pairwise similarity function S_f is defined as in Eq. (C.1) with a bandwidth value $b = 0.5$. For each text query ϕ_q, we define a unary regularization term P_q using a logistic regression model $\mathcal{L}_q$ that predicts the probability that a DINOv2 feature f_i belongs to the object corresponding to query ϕ_q. The set of nodes $\mathcal{P}$ with positive labels is defined based on 3D CLIP relevancy scores for prompt ϕ_q. More precisely, we rescale 3D relevancies to $[0, 1]$ and apply Otsu's method (Otsu et al., 1975) over relevancies above 0.5. We use a regularization $C = 0.001$ and equal class weighting for training the model. The unary regularization term P_q is then defined as $P_q(f_i) = \mathcal{L}_q(f_i)^{1/b}$, with $b = 0.025$ for segmentation with SAM, and $b = 2$ otherwise. The initial weight vector g_0 is defined by applying two more iterations of Otsu's method among nodes in $\mathcal{P}$ and setting to zero all relevancies below the given threshold. Restricting the set of initial points ensures diffusion only happens within the object of interest. Segmentation based on the resulting 3D relevancy scores is then performed as described in the previous paragraphs, using $\tau = 0.01$ for segmentation with SAM and $\tau = 0$ otherwise.

C.2 Additional results

C.2.1 Runtime analyses

In this section, we detail our running times for feature uplifting and evaluation, conducted on a GPU A6000 ADA. Table C.1 shows a breakdown of running times between feature uplifting (Up.) and generation (Gen.), graph diffusion and 2D segmentation for evaluation on LERF segmentation. The total reported times can be divided between pre-uplifting, uplifting and post-uplifting. In our experiments, the pre-processing and uplifting steps are independent from the downstream tasks (except for our foreground/background segmentation with SAM), and part of the graph-diffusion process is task-dependent. Below we detail our runtimes for every step and compare them to the literature.

Pre-uplifting: feature map generation. The time this step takes (Gen. in Table C.1) depends on the backbone model, on the number of images and on the number of calls to the model per image. The total time ranges from a few seconds up to an hour for approaches such as LangSplat (Qin et al., 2024), that queries SAM over a grid of points on the image at various resolutions to generate full image segmentation masks. This process takes 24s/image on a GPU 6000 ADA and amounts to an average of 80 minutes for the evaluated scenes. In our experiments, the feature generation step takes from 1 to 5 minutes.

Uplifting. For LUDVIG, uplifting time is linear in the number of images (given a 3D scene representation). Experimentally, it is also linear in the number of feature dimensions, taking 2ms per dimension for an image of size 724×986. As reported in Table C.1 (Up.), uplifting 100 images of dimension 40 takes 9s on average. By contrast, gradient-based optimization requires approximately n_{steps} times this duration, where the number of gradient steps n_{steps} typically ranges from 3,000 to 30,000 for 3D feature distillation (Kerr et al., 2023; Qin et al., 2024; Zuo et al., 2024). Gradient-based optimization can still be very fast for low-dimensional or dimensionality-reduced features. For example, SAM masks can be optimized in as little as a few seconds, as reported by SA3D-GS (Cen et al., 2023b). LangSplat (Qin et al., 2024) trains an autoencoder to reduce the CLIP feature dimensionality from 512 to 3, and runs in 25 minutes. However, optimization becomes intractable for high-dimensional features such as CLIP and DINO; FMGS (Zuo et al., 2024) relies on an efficient multi-resolution hash embedding of the scene; however, their total training time still amounts to 1.4 hours.

Post-uplifting: graph diffusion. After uplifting, LUDVIG performs graph diffusion using pairwise DINOv2 feature similarities for segmentation tasks. In Table C.1, we divide runtimes in two categories:

- **Scene**: operations performed once for the whole scene. This includes querying the Euclidean neighbors for each node, which is log-linear in the number of Gaussians. With 600,000 Gaussians as in our experiments, the step takes about 30s, and can be further optimized by using approximate nearest neighbor search algorithms (Wang et al., 2021a). Defining S_f based on DINOv2 features is also independent from the downstream task.
- **Prompt**: operations that are specific to the downstream task. This includes defining the regularization P (e.g. training logistic regression model(s)) and running the diffusion. The time taken depends on dimension of the diffused features (e.g. number of text queries): 1 to 2 seconds for foreground/background segmentation (a single mask) and 18 seconds on average for LERF segmentation (14 to 21 text queries).

Post-uplifting: segmentation. Our evaluation on LERF involves 2D segmentation with SAM based on 3D relevancy scores. The runtime depends on the number of test images and on the total number of 2D text queries, as it involves one call to the SAM backbone per test image, and multiple calls to the SAM prediction head per text query, as detailed in Appendix C.1.4. Our total inference time per scene is of $8s$ on average, against $0.8s$ when not using SAM. In contrast, Langsplat does not

	Images (#)		**Text queries (#)**		**DINOv2 (s)**		**CLIP (s)**	**Graph diffusion (s)**		**2D segmentation (s)**		Total
Scene	Train	Test	Unique	Total	Gen.	Up.	Gen.+Up.	Scene	Prompt	w/ SAM	w/o SAM	(mins)
Teatime	177	6	14	59	45	14	363	42	15	9	0.9	8
Waldo	187	5	18	22	44	18	371	39	19	4	0.5	8
Ramen	131	7	14	71	40	9	227	37	14	11	1	6
Figurines	299	4	21	56	58	38	811	45	22	8	0.8	16

TABLE C.1: **Runtimes for evaluation on LERF segmentation (Kerr et al., 2023; Qin et al., 2024).** The last column (Total) reports total time, which breaks down between i) feature map generation (Gen.) and uplifting (Up.) for all training images; ii) graph diffusion, divided between scene-specific (querying neighbors, defining S_f) and prompt-specific (defining P, running diffusion) operations for all text queries; iii) 2D segmentation with/without SAM for all text queries across test images. We also report the number of training and test images and the number of text queries across test images.

	MVSeg	SA3D-GS	SAGA	OmniSeg3D	LUDVIG (Ours)		
3D representation:	NeRF	GS	GS	NeRF	GS		
Uplifting:		SAM	SAM	SAM	DINOv2	SAM	SAM2
Orchids	92.7	84.7	-	92.3	92.6	92.2	91.0
Leaves	94.9	97.2	-	96.0	96.2	96.3	96.3
Fern	94.3	96.7	-	97.5	96.3	97.0	97.0
Room	95.6	93.7	-	97.9	95.7	96.5	96.1
Horns	92.8	95.3	-	91.5	95.1	94.5	94.8
Fortress	97.7	98.1	-	97.9	97.5	98.3	98.3
Fork	87.9	87.9	-	90.4	85.0	86.8	86.7
Pinecone	93.4	91.6	-	92.1	93.2	88.8	90.7
Truck	85.2	94.8	-	96.1	95.6	94.9	93.9
Lego	74.9	92.0	-	90.8	91.1	92.7	92.9
Average	90.9	93.2	93.4	94.3	93.8	93.8	93.8

TABLE C.2: Segmentation (IoU) on SPIn-NeRF (Mirzaei et al., 2023) with DINOv2, SAM and SAM2.

	Fern	Flower	Fortress	HornsC	HornsL	Leaves	Orchids	Trex	Average
NVOS	-	-	-	-	-	-	-	-	70.1
SA3D	82.9	94.6	98.3	96.2	90.2	93.2	85.5	82.0	90.3
OmniSeg3D	82.7	95.3	98.5	97.7	95.6	92.7	84.0	87.4	91.7
SA3D-GS	-	-	-	-	-	-	-	-	92.2
SAGA	-	-	-	-	-	-	-	-	92.6
Ours-DINOv2	84.5	95.6	97.5	97.3	93.4	96.3	91.7	84.7	92.4
Ours-SAM	85.5	97.6	98.1	97.9	94.1	96.4	73.1	88.0	91.3
Ours-SAM2	84.8	97.3	98.3	97.7	93.4	96.7	73.1	89.1	91.3

TABLE C.3: Segmentation (IoU) on NVOS (Ren et al., 2022) with DINOv2, SAM and SAM2.

rely on SAM at inference time, but relies on a computationally expensive feature map generation process, with more than 1 hour runtime.

	Geometry only	Single view		Uplifting		Graph diffusion
Model:	Reference mask	DINOv2	SAM2	DINOv2	SAM2	DINOv2
Orchids	71.3	91.5	78.4	91.5	91.0	92.6
Leaves	72.4	89.3	96.6	94.1	96.3	96.2
Fern	93.9	95.1	96.7	96.7	97.0	96.3
Room	77.4	95.4	95.6	97.3	96.1	95.7
Horns	80.7	90.9	93.0	94.2	94.8	95.1
Fortress	94.3	96.8	97.7	98.6	98.3	97.5
Fork	67.5	85.6	80.5	88.8	86.7	85.0
Pinecone	56.5	92.8	67.8	89.6	90.7	93.2
Truck	60.1	83.6	90.9	95.2	93.9	95.6
Lego	57.3	64.4	89.0	69.9	92.9	91.1
Average	73.1	88.5	88.6	91.6	93.8	93.8

TABLE C.4: **Segmentation (IoU) on SPIn-NeRF (Mirzaei et al., 2023)**. We compare purely geometrical reference mask uplifting and reprojection, single-view prediction, feature/-mask uplifting, and graph diffusion leveraging DINOv2 or SAM2.

C.2.2 Per-scene foreground/background segmentation results

In this section, we present per-scene segmentation results on NVOS and SPIn-NeRF in Tables C.2, C.3 and C.4, along with an extended analysis of these results.

Segmentation on SPIn-NeRF. We report our segmentation results for the SPin-NeRF dataset (Mirzaei et al., 2023) in Table C.2. Our results are comparable to the state of the art while not relying on optimization-based approaches. Surprisingly, our segmentation with DINOv2 using graph diffusion also gives results on par with models leveraging SAM masks. Our lower segmentation results compared to OmniSeg's can be partly attributed to poor Gaussian Splatting reconstruction of highly specular scenes such as the Fork, in which semi-transparent Gaussians floating over the object try to represent reflections or surface effects that are difficult to capture with standard rasterization techniques (Jiang et al., 2024).

Segmentation on NVOS. We report our segmentation results for the NVOS dataset (Ren et al., 2022) in Table C.3. Our results are comparable to those obtained by prior work. Again, DINOv2 performs surprisingly well while not having been trained on billions of labeled images like SAM. Compared to SAM, DINOv2 better captures complex objects, but sometimes also captures some background noise. This can be seen in Appendix Fig. C.2 with the example of Trex: while SAM misses out the end of the tail as well as the end of the ribs, DINOv2 captures the whole Trex, but also captures part of the stairs behind. Visualisations of Orchids in Appendix Fig. C.2 also explain the lower performance of SAM on this scene: the two orchids SAM is missing are not covered by the positive scribbles, which makes the task ambiguous.

Ablation study. In Table C.4, we compare our segmentation protocol using DINOv2 and SAM2 to multiple simpler variants. More precisely, we evaluate i) a purely geometrical variant that does not use SAM2 or DINOv2, ii) single-view segmentation in 2D based on SAM2 or DINOv2 2D predictions, iii) uplifting DINOv2 features or SAM2 masks into 3D then rendering them for segmentation, as described in Sec. C.1.1 and C.1.2, and iv) segmenting using graph diffusion over DINOv2 3D feature similarities.

The purely geometrical approach works well on the forward-facing LLFF scenes (Orchids to Fortress). In these scenes, the reference mask is accurately uplifted and reprojected as the viewing direction changes only a little between each frame. However, it fails on the 360-degree scenes (Fork, Pinecone, Truck, Lego). This points to a suboptimal 3D reconstruction of the scene, likely due to overfitting on the limited numbers of available views (Chung et al., 2024).

The single-view variants use a similar process for constructing the features and using them for segmentation as in Sec. C.1.1 and C.1.2 but without uplifting and rendering. It improves from a purely geometrical approach and performs reasonably well on average, the foreground being well isolated from the rest of the scene. However, as illustrated in Fig. 3, the semantic features are at a much lower resolution than those resulting from 3D uplifting, leading to a coarser segmentation.

3D uplifting considerably boosts results compared to single-view approaches. However, performing segmentation in 2D based on the uplifted DINOv2 features does not benefit from the 3D spatial information and typically fails on the 360-degree scenes (Pinecone, Truck and Lego) which have higher variability between frames from different views. Introducing 3D spatial information through 3D graph diffusion results in a boosted performance on these scenes.

C.3 Additional visualizations

C.3.1 Segmentation tasks

Segmentation on NVOS. Fig. C.2 shows our segmentation masks from SAM and DINOv2 for the three most challenging scenes of the NVOS dataset: Fern, Orchids and Trex.

Diffusion process. Fig. C.3 illustrates different steps of the diffusion process for Fern, Leaves, Flower and Trex from the NVOS (Ren et al., 2022) dataset. Starting from the reference scribbles, the diffusion rapidly spreads through the large neighboring Gaussians. Covering the entire object takes more time for complex structures such as

(A) Reference image (B) Ground truth mask (C) DINOv2 mask (D) SAM mask

FIGURE C.2: Segmentation results on NVOS (Ren et al., 2022) with DINOv2 and SAM.

Fern, or for masks with disconnected components such as Orchids. As illustrated in the case of Flower, the last diffusion steps allow spreading to the smaller Gaussians on the flowers' edges, yielding a refined segmentation mask. For Trex, the parts being reached the latest are the head and tail. Their features are further away from the reference features (defined as the average feature over 3D reference scribbles), and therefore the regularization for diffusion is stronger in these regions. Overall once the object has been fully covered, the regularization is very effective at preventing leakage, which allows diffusion to run for an arbitrary number of steps.

Object removal. Fig. C.4 shows comparative visualizations of object removal with N3F (Tschernezki et al., 2022) and LUDVIG. For rendering the edited RGB image, N3F sets to zero the occupancy for all 3D points belonging to the object. For LUDVIG, we remove all Gaussians pertaining to the 3D semantic mask resulting from graph diffusion. We observe that the regions behind to segmented object are much smoother for LUDVIG than for N3F. Regions unseen from any viewpoint are black for LUDVIG (no gaussians) and result in a background partially hallucinated by NeRF for N3F.

C.3.2 Visual comparisons of uplifted features

Fig. C.5 show a comparison of LUDVIG's 3D DINOv2 features with learned 3D DINO features of N3D (Tschernezki et al., 2022). Their figures are taken from their work.

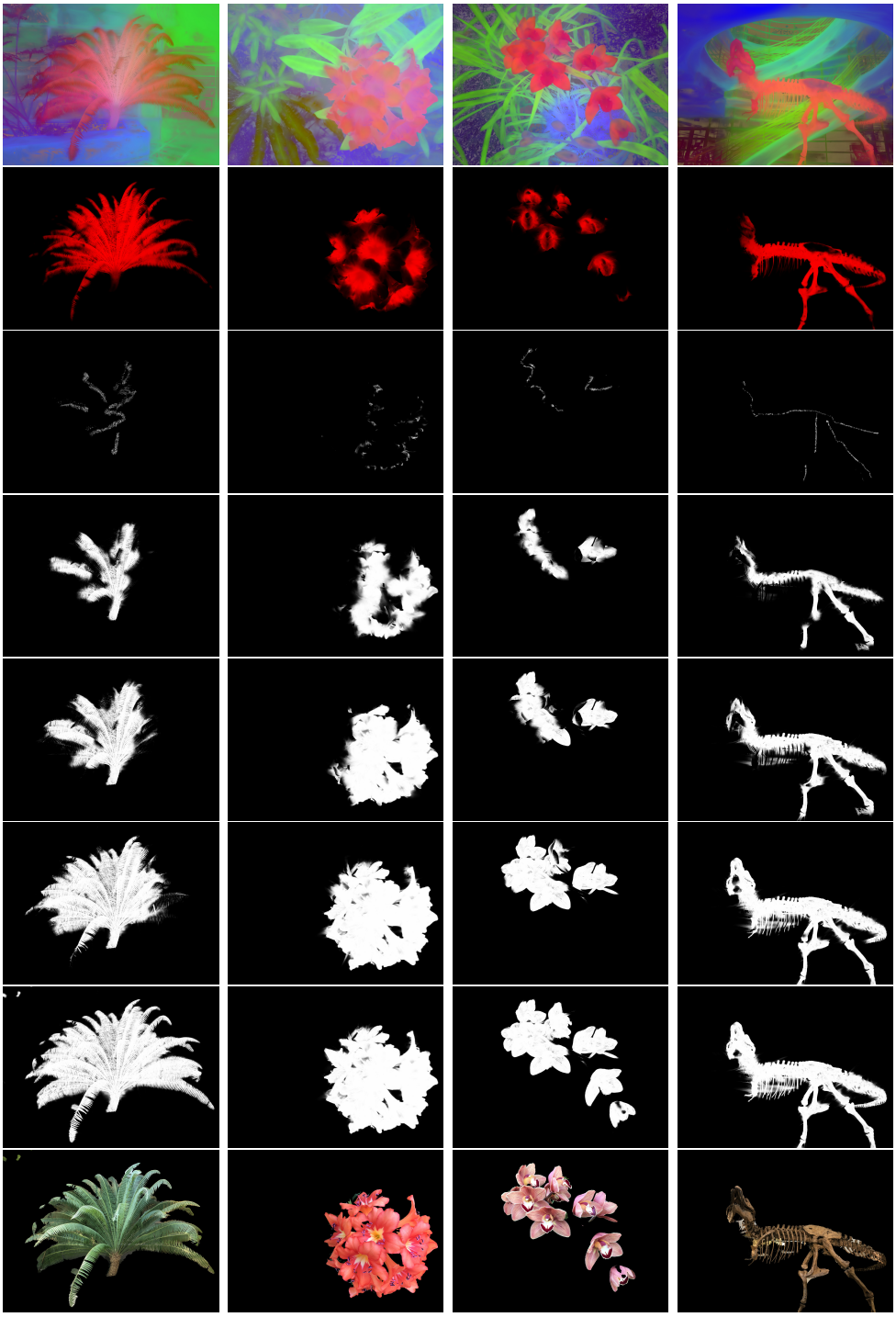
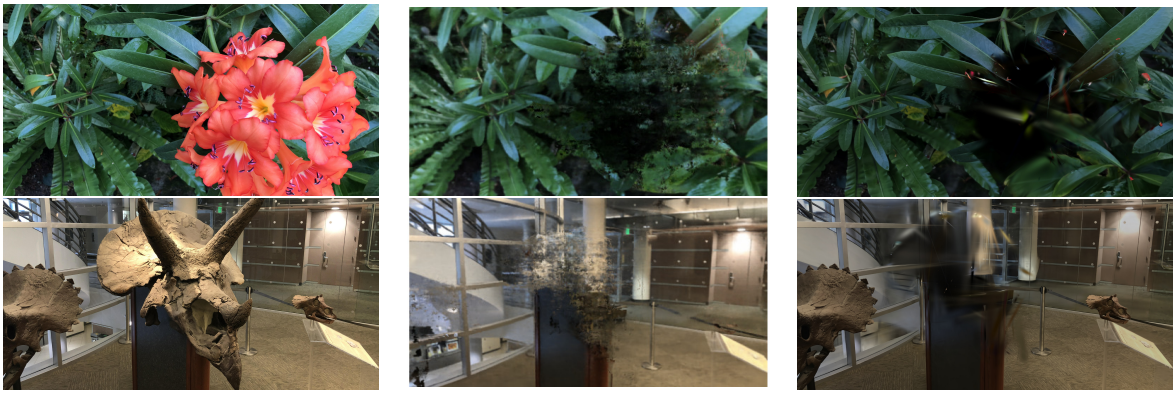


FIGURE C.3: **Illustration of the graph diffusion process.** 2D projections of i) first three PCA components of DINOv2 3D features, ii) unary regularization term (red), iii) weight vector g_t at timesteps $t \in \{0, 3, 5, 10, 100\}$, iv) RGB segmentation obtained using a mask based on the 2D projection of g_{100}.

The notable differences are a more fine-grained reconstruction of the background for the trex and horns, and overall smoother features across all scenes.



(A) RGB image (B) N3F (Tschernezki et al., 2022) (C) LUDVIG (ours)

FIGURE C.4: **Object removal.** 3D segmentation, removal and rendering for LUDVIG and N3F (Tschernezki et al., 2022). For N3F, figures are sourced from (Tschernezki et al., 2022).

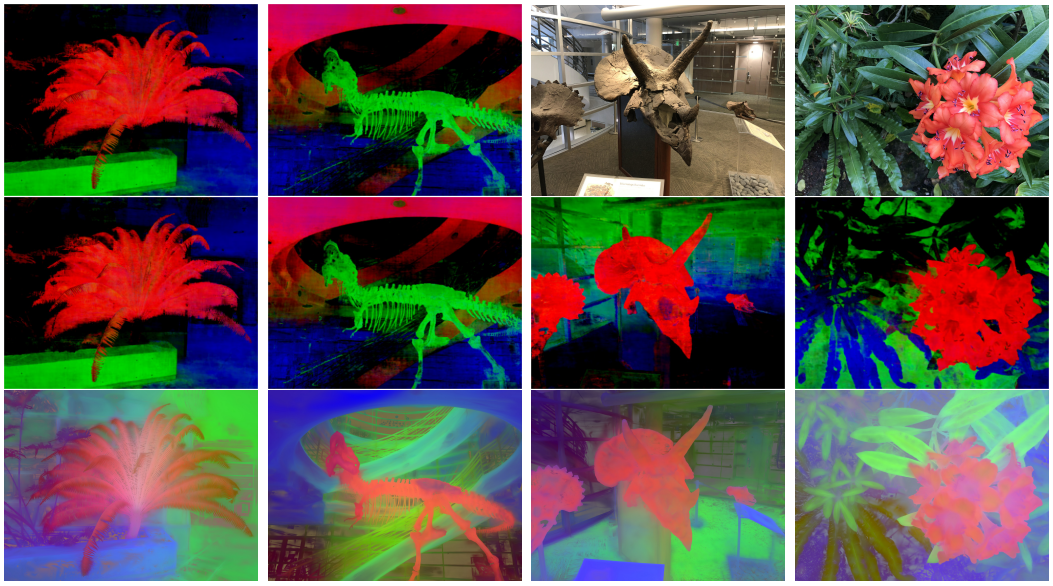
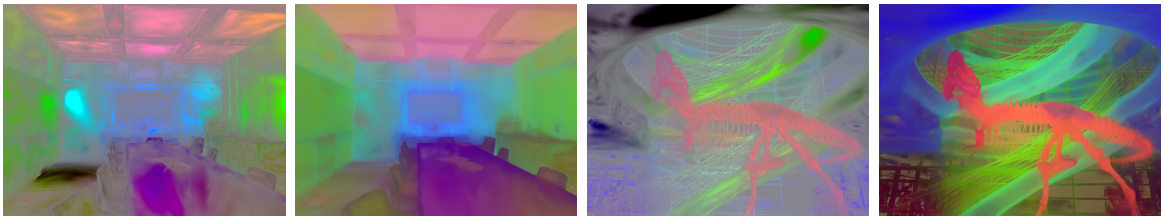


FIGURE C.5: Comparison between LUDVIG's uplifted DINOv2 features (bottom) and N3F's (Tschernezki et al., 2022) learned DINO features (top). For N3F, figures are sourced from (Tschernezki et al., 2022).

C.3.3 Comparison to GaussianEditor's uplifting.

Our aggregation procedure in Eq. (5.3) from the main paper, illustrated in Fig. 2, bears similarity with the one from Chen et al. (2024) for uplifting 2D binary masks to a 3D Gaussian splatting scene. In their method, uplifted masks are thresholded to create 3D binary masks that are used for semantic tracing. Specifically, they rely on rough 3D segmentation masks to selectively optimize Gaussians that are relevant for an editing task. Unlike in Eq. (5.3) and (5.5) from the main paper, Chen et al. (2024) propose to normalize their uplifted masks based on the total count of view/pixel pairs (d, p) contributing to the mask of a Gaussian i, i.e. $\sum_{d,p \in \mathcal{S}_i} 1$, without taking the rendering weight $w_i(d, p)$ into account. Consequently, the uplifted features tend to have larger



(A) GaussianEditor (B) LUDVIG (C) GaussianEditor (D) LUDVIG

FIGURE C.6: **Comparison to GaussianEditors's uplifting.** Comparison of PCA visualization of uplifted features between LUDVIG's and GaussianEditor's aggregation (Chen et al., 2024).

values for large, opaque Gaussians. Fig. C.6 shows a qualitative comparison between 3D DINOv2 features obtained using the aggregation proposed by Chen et al. (2024) and our approach. The aggregation by Chen et al. (2024) fails to assign the right semantics to large gaussians, which is particularly visible in scenes with high specularity such as Room. This showcases the importance of defining 3D features as convex combinations of 2D pixel features.

C.3.4 Visualization of CLIP segmentation results

In this section, we present illustrations of the impact of the diffusion process (Fig. C.7), and comparative visualizations of localization heatmaps with LangSplat and LERF (Fig. C.8).

C.3.4.1 Impact of segmentation with SAM and DINOv2-guided graph diffusion

Fig. C.7 shows 2D segmentation masks colored by CLIP relevancy scores, obtained with and without leveraging SAM and/or DINOv2-guided graph diffusion for refining 3D relevancy scores.

Direct segmentation from raw 3D relevancy scores. Isolating a specific object in the scene directly based on CLIP relevancy scores is challenging: the segmentation masks obtained by automatic thresholding include parts of other objects with similar features, like for the sheep (segmentation of the bear nose) and the spoon. The segmentation might also cover surroundings of the object of interest simply due to the low resolution of CLIP visual features, such as in the knife example.

2D segmentation with SAM. SAM delivers a precise 2D segmentation of the object covered by points with the highest relevancy scores. However, point prompts may not span the entire object, resulting in undersegmentation, like for the sheep. In some cases, point prompts with the highest relevancy may even be located on the wrong

object, resulting in an entirely wrong segmentation (e.g., the bowl segmented instead of the spoon).

Relevancy score refinement with graph diffusion based on 3D DINOv2 features. The graph diffusion process starts with positive weights for Gaussians with the highest relevancy scores, and propagates their weight to neighbors with similar DINOv2 features. However in cases where the object of interest consists of multiple subparts (e.g. for the sheep), the final distribution of weights may be inhomogeneous and the automatic thresholding may select only one subpart. Also, if multiple close objects are to be segmented (e.g. with the knife), the final weights may cover surrounding Gaussians and the final thresholding might not clearly isolate the objects.

3D graph diffusion combined with 2D SAM segmentation. Combining 3D graph diffusion and 2D SAM segmentation helps solving the aforementioned problems observed when using either of the two approaches individually. The diffusion process allows selecting a large set of point prompts for SAM spanning the object of interest without covering other object with similar features, resulting in an accurate segmentation.

C.3.4.2 Qualitative comparison of open-vocabulary objet localization.

Fig. C.8 illustrates open-vocabulary object localization with LERF (Kerr et al., 2023), LangSplat (Qin et al., 2024) and LUDVIG. Both LangSplat and LUDVIG correctly localize all four example objects. For queries such as the chopsticks, LangSplat's localization is more precise, as the CLIP features are constructed by generating full image segmentation masks with SAM. This process is computationally expensive, as constructing a full segmentation mask requires querying SAM over a grid of points on the image and takes about 23s for a single image (on a GPU A6000 ADA), which amounts to an average of 80 minutes for a scene from the LERF dataset. However, it yields coherent instance-level CLIP representations, which is desirable for downstream segmentation tasks.

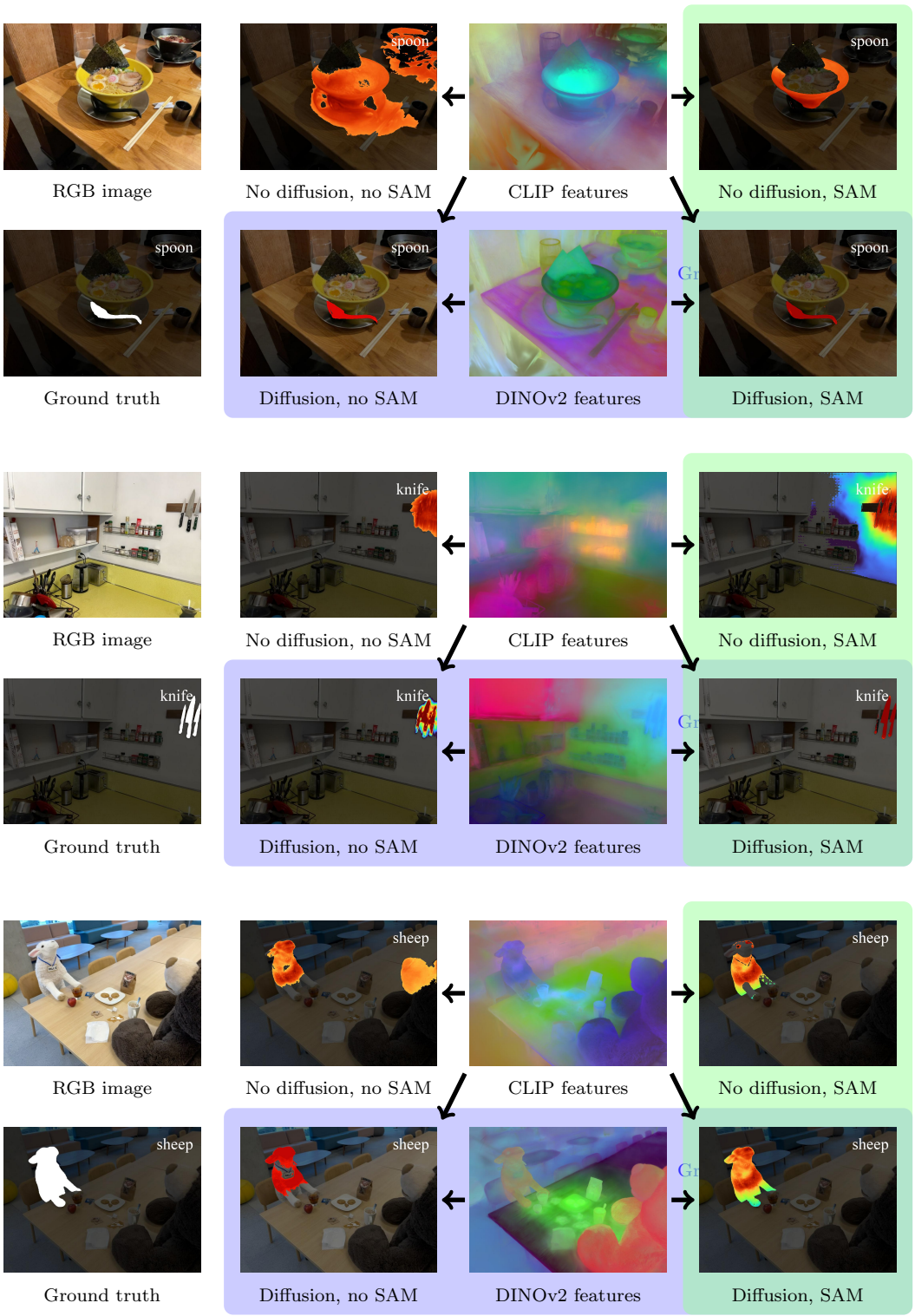


FIGURE C.7: **Open-vocabulary object segmentation with and without using 3D graph diffusion (blue) and/or 2D SAM segmentation (green).** Projections of 3D CLIP and DINOv2 features colored by three main PCA components and 2D segmentation masks colored by relevancy scores.

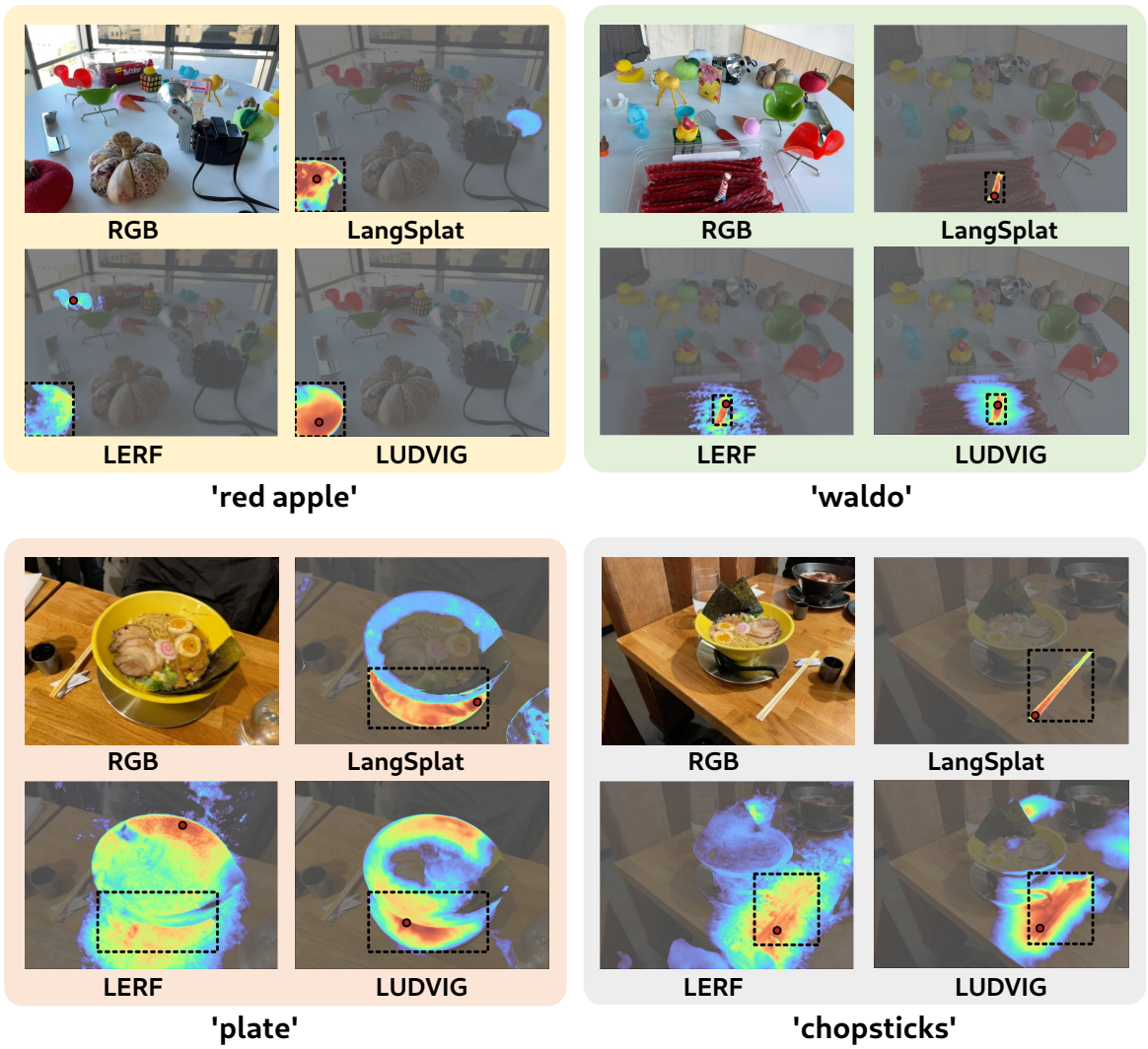


FIGURE C.8: **Qualitative comparisons of open-vocabulary 3D object localization on the LERF dataset.** The red points are the model predictions and the black dashed bounding boxes denote the annotations. This figure is sourced and adapted from LangSplat's website (https://langsplat.github.io/), licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.