

Alma Mater Studiorum - Università di Bologna

DOTTORATO DI RICERCA IN
AUTOMOTIVE PER UNA MOBILITÀ INTELLIGENTE

Ciclo 36

Settore Concorsuale: 09/H1 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

Settore Scientifico Disciplinare: ING-INF/05 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

ENHANCING AUTONOMOUS DRIVING VISION: DEEP LEARNING
APPROACHES TO DATA SYNTHESIS, 3D REPRESENTATION AND MODEL
EFFICIENCY

Presentata da: Anthony Dell'Eva

Coordinatore Dottorato

Nicolò Cavina

Supervisore

Massimo Bertozzi

Co-supervisore

Stefano Selleri

Esame finale anno 2024

*To my parents
and friends*

Abstract

In the autonomous driving domain, effectively perceiving and interpreting the environment is crucial for safe and efficient navigation. The field has been revolutionized by deep learning models that enable the vehicle to detect, recognize and segment objects in the surrounding environment. However, several challenges persist, including the time-consuming process of data collection and labeling, inconsistencies in the quality of input data, and the constraint of real-time processing. To address these issues, we present three approaches: the augmentation of training datasets, the refinement of input data during the inference stage, and the transfer of knowledge from complex to lightweight models via a process known as knowledge distillation. We show that generative adversarial networks and attention-based architectures can be leveraged, respectively, to generate realistic samples from a learned data distribution and to increase the input resolution. These methods can improve the generalization capabilities of detection, segmentation, and reconstruction algorithms. Furthermore, a significant contribution of our research is the integration of knowledge distillation, aligning the performance of simpler models with their more complex counterparts. This ensures the perception systems are not only accurate but also versatile in real-time scenarios, a critical aspect for the successful deployment of safe autonomous vehicles.

Contents

1	Introduction	1
1.1	Autonomous Driving	1
1.2	Deep Learning for Vehicle Perception	6
1.2.1	Perception Tasks	8
1.3	Research Objectives and Motivation	13
1.3.1	What does it mean to enhance a deep learning model? . .	13
1.3.2	How to enhance a deep learning model?	14
1.3.3	Contributions	15
2	Adversarial Networks for Synthetic Data Generation	17
2.1	Introduction to GANs	17
2.1.1	Training and Evaluation	19
2.1.2	Network Architecture	21
2.1.3	Limitations	22
2.1.4	Applications	22
2.2	Image-to-Image Translation	23
2.2.1	Supervised Image Translation	24
2.2.2	Unsupervised Image Translation	24
2.3	Study Objective and Contributions	25
2.4	Related works	27
2.5	Method	29
2.5.1	Local domains	29
2.5.2	Geometrically-guided patches	30
2.5.3	Local domains interpolation	31
2.5.4	Training	32
2.6	Experiments	32
2.6.1	Tasks definitions	34

CONTENTS

2.6.2	Evaluation	35
2.6.3	Ablation study	40
2.7	Supplementary results	42
2.7.1	GAN metrics	42
2.7.2	Additional qualitative results	44
2.8	Discussion	45
3	Point Cloud Upsampling for Enhanced 3D Data	47
3.1	2D vs. 3D Data in Autonomous Driving	47
3.2	3D Data Representations	48
3.3	Study Objective and Contributions	51
3.4	Related Work	53
3.5	Method	55
3.5.1	Feature Extraction	56
3.5.2	Spherical Mixture of Gaussians Sampling	56
3.5.3	Details on SMOG Sampling	57
3.5.4	Transformer Model	59
3.5.5	Residual Refinement	59
3.5.6	Loss Function	60
3.6	Experiments	62
3.6.1	Settings	62
3.6.2	Quantitative Results	63
3.6.3	Qualitative Results	64
3.6.4	Generalization and Robustness	65
3.6.5	Surface Reconstruction	66
3.6.6	Ablation Studies	68
3.6.7	Robustness to Input Size	70
3.6.8	Additional Qualitative Results	71
3.7	Discussion	71
4	Knowledge Distillation for Model Efficiency	75
4.1	Study Objective and Contributions	75
4.2	Related Work	78
4.3	Method	80
4.3.1	Preliminaries	80
4.3.2	Teacher Features-Driven Regularization	81

4.3.3	Sparse Self-Distillation	82
4.4	Experiments	83
4.4.1	Classification	83
4.4.2	Object Detection and Instance Segmentation	85
4.4.3	Semantic Segmentation	85
4.4.4	Ablations	87
4.4.5	Analysis	88
4.4.6	Sparse Self-Distillation	93
4.5	Discussion	95
5	Conclusions	97
A	Overview of Deep Learning Technologies	101
A.1	Generative vs. Discriminative Models	101
A.2	Architectures	102
A.2.1	Convolutional Neural Networks	102
A.2.2	Transformers	104
	References	109



Introduction

1.1 AUTONOMOUS DRIVING

In recent years, the evolution of self-driving technologies has experienced dramatic exponential advancements. This growth, driven by industries and research teams, has drawn more focus from institutions and the media, accordingly. Unfortunately, nowadays the field is facing a notable lack of government regulations, which slows down the technology progress instead of boosting it. The media, on the other hand, is vital in raising awareness about self-driving cars in public opinion, helping people familiarize themselves with the related implications this revolution could bring on different levels. This includes making mobility more efficient, improving road safety, and offering increased productivity and new business opportunities [240].

Impact and Challenges. Autonomous driving (AD) has the potential to reshape the mobility sector due to key trends like electrification, ride-sharing, and connectivity. The popularity of electrification is rising because of cheaper batteries and government incentives, and autonomous vehicles (AVs) predominantly being electric will boost this trend. Ride-sharing giants like Uber and Lyft challenge the concept of car ownership, and the advent of AVs promises more affordable, 24/7 ride-sharing services. Connectivity will be enhanced by AVs communicating with each other to enhance traffic flow and safety. The rise of autonomous driving offers significant social and economic benefits [310, 22, 127, 218], including increased productivity and entertainment for passengers. No-

1.1. AUTONOMOUS DRIVING

tably, AVs can dramatically reduce the global annual 1.3 million deaths from car accidents. AVs offer greater mobility for the elderly and disabled [274], potential cost savings from accidents, and environmental benefits. A study indicates AVs could be 25% more fuel-efficient than human-driven cars [82], and optimized parking could save up to 15% of space [15]. Furthermore, industries like mining, farming, and construction are integrating automation, with trucking seeing significant advancements (see, for example, the Waymo Via project).

Unfortunately, driverless mobility faces various challenges [22]. One primary concern is ensuring the safety of the technology, especially given the lack of comprehensive data to demonstrate its reliability compared to the fatality rate. Moreover, while AVs might surpass human driving capabilities, it is uncertain if they are superior to humans using advanced driver-assistance systems (ADAS), like adaptive cruise control and lane-keeping. Ethical dilemmas also arise, such as prioritizing the safety of passengers versus pedestrians in emergencies [274]. Cybersecurity threats, like hacking, pose risks to public safety, and outdated government regulations and inadequate infrastructure, like road potholes and poor lane markings, hinder AV integration. Finally, while there are concerns about massive job losses in sectors like taxi and delivery services, it is believed many will be reemployed in transportation, AV-related fields, and new services developing from emerging business models and consumer habits [219].

History of Vehicle Automation. The history of vehicle automation began around a century ago with Francis Houdina's creation of a radio-controlled car in 1925 [81]. This car, modified from a Chandler Sedan, could drive through Manhattan streets without a driver, steered remotely via radio signals. By the 1950s, the Radio Corporation of America (RCA) experimented with guide wires embedded in roads to control vehicles, but the costs were prohibitive [1]. The focus then shifted in the 1960s to autonomous droids at Stanford University that used cameras and computers to navigate [156]. With advancements in CPUs and computer vision, progress accelerated. In 1977, the University of Tsukuba developed a self-driving car that used cameras to detect road markings [18]. By 1986, German engineer Ernst Dickmanns and his team tested a self-driving Mercedes van, and later, a Mercedes 500 SEL that could execute complex maneuvers [72]. In 1995, Carnegie Mellon University showcased a modified Pontiac Trans Sport that drove autonomously for most of a cross-country trip, pioneering the use of GPS in such vehicles [249]. Italy also saw advancements: in 1996, the

University of Parma ARGO project enabled a Lancia Thema to autonomously navigate Italian highways [25]. Between 2004 and 2007, the Defense Advanced Research Projects Agency (DARPA) held three autonomous driving challenges, with Stanford University and the Tartan Racing team emerging as winners in separate events [12, 303, 311]. In 2010, VisLab (University of Parma) completed a 9900-mile intercontinental autonomous drive [27]. Recent developments in artificial intelligence have further enhanced the field. Currently, companies are increasingly attempting to introduce autonomous vehicles to the general public, but a fully self-driving car does not yet exist. Despite the enormous hype surrounding AV technology, interest appears to be slowly diminishing. However, this decrease in attention doesn't imply that the autonomous revolution will not take place or that full automation is not feasible. Instead, it suggests that the technology is not yet mature and that the real benefits and impacts are not yet completely understood. It might take decades before AVs substantially influence the mobility and transportation industries and our daily lives.

Automation Levels. The Society of Automotive Engineers (SAE) categorizes vehicle autonomy into six levels. Level 0 refers to *no automation*, where the vehicle has no self-driving capabilities and the human driver manages all tasks. Level 1, known as *driver assistance*, allows the car to assist with either steering or braking, but not both simultaneously, with adaptive cruise control being a typical feature. Level 2, or *partial automation*, lets the vehicle handle functions like acceleration and steering, but it still requires full driver attention. Tesla's Autopilot and GM's Super Cruise are examples of this level. Level 3, named *conditional automation*, permits the vehicle to manage more complex tasks, with the driver intervening only in emergencies. Audi A8 was the pioneer in achieving this level. Level 4, *high automation*, allows the car to undertake all driving tasks in specific scenarios such as in designated zones and during favorable weather, although drivers have the option to take control. Lastly, Level 5 represents *full automation*, where the vehicle can operate in any condition without human intervention.

Currently, we have achieved level 3 autonomy, as seen in vehicles like the Audi A8. Several production vehicles offer level 2 (or level 2+) advanced automation, such as Tesla, while companies including Waymo and Zoox are testing level 4 autonomous vehicles on U.S. roads.

1.1. AUTONOMOUS DRIVING

Hardware. The sensory systems of autonomous vehicles can be grouped into three primary categories: self-sensing, localization and surrounding sensing. Self-sensing involves tracking the vehicle’s current state, like velocity, acceleration, yaw, and steering angle, using internal sensors such as odometers, IMUs, gyroscopes, and the CAN bus. Localization involves determining the vehicle’s global and local position using external sensors like GPS or dead reckoning by IMU readings. Surrounding sensing employs sensors to perceive the external environment.

The key sensors include cameras, radar, sonar, and LiDAR. Cameras can sense color and are passive, i.e. they do not emit any signal for measurements. They can distinguish objects by texture and contrast. However, they are limited in challenging light or weather conditions and do not provide distance information without additional setups or software [17, 16, 173]. Traditional cameras typically consist of an aperture and one or several lenses, and they can be effectively approximated by the pinhole camera model. Omnidirectional cameras use mirrors or special lenses to significantly expand the field of view, while event cameras capture changes in intensity with very high temporal resolution. Radars use electromagnetic waves to measure distances and are especially reliable for short distances, but their resolution is low [71, 294]. Sonars operate using sound waves and are primarily used for close-range tasks, such as parking assistance [139]. LiDAR sensors, which emit laser pulses, create detailed 3D maps of the environment, allowing for the recognition and classification of objects. Recently, they have become more affordable due to advancements in solid-state design [180, 179, 85]. Given the strengths and weaknesses of each sensor, vehicles typically incorporate multiple types. This diverse data is then integrated through *sensor fusion* to create a precise representation of the environment around the vehicle [151, 349]. Nowadays, many industry players are developing systems that use cameras and radars but rely on LiDAR as the primary sensor. On the other hand, companies like Tesla prefer more cost-effective solutions based exclusively on cameras and ultrasonic sensors.

Finally, the computing platform plays a crucial role. It supports sensor data processing to understand surroundings and control vehicles using advanced algorithms of optimization, computer vision, and machine learning. Various computing solutions exist, including CPUs, GPUs, ASICs and FPGAs.

Software. As stated previously, autonomous vehicles make decisions by processing data from several onboard sensors. This data is used by the vehicle's computational system to take driving actions. Fig. 1.1 presents basic block diagrams of a self-driving car. The vehicle decisions are computed through a modular perception-planning-action sequence, as depicted in Fig.1.1a, or via a comprehensive end-to-end learning approach, illustrated in Fig. 1.1b, where sensory data is directly translated to vehicular controls.

The main components of the modular pipeline are the following:

- **Perception.** This module processes sensor data to understand the environment around it. The perception module identifies features to gather information about lane markings [377, 230], curbs [365], and traffic signs [205, 296]. It also determines where the car can safely drive. Additionally, it can detect and classify objects [48, 327] on the road, understanding details like their size, position, and movement. By tracking these objects over time, the system can predict their future actions [200].
- **Localization and Mapping.** This module primarily aims to determine the vehicle's position [158, 170] and create a model of the surrounding environment [276]. When both these tasks are done together, especially when sensor positions are uncertain, it is referred to as Simultaneous Localization and Mapping, or SLAM [297]. This ensures the vehicle recognizes its location and stays in the correct lane. It also helps the vehicle navigate without colliding by understanding the position of nearby objects. It is crucial for these estimates to be precise, as even minor mistakes can lead to severe outcomes. A valuable resource in this context is HD maps (High Definition Maps) [11]. These maps offer detailed information, with accuracy down to centimeters, allowing vehicles to navigate with fewer errors.
- **Planning.** Motion planning, also known as path planning [302, 54, 162, 161], involves determining a sequence of states that enables a system to transition from an initial to a target position, while avoiding potential obstacles. In the context of autonomous driving, this planning must take into account specific constraints, including speed limits. The effectiveness of this module is conditioned on the precision of data provided by previous components.

1.2. DEEP LEARNING FOR VEHICLE PERCEPTION

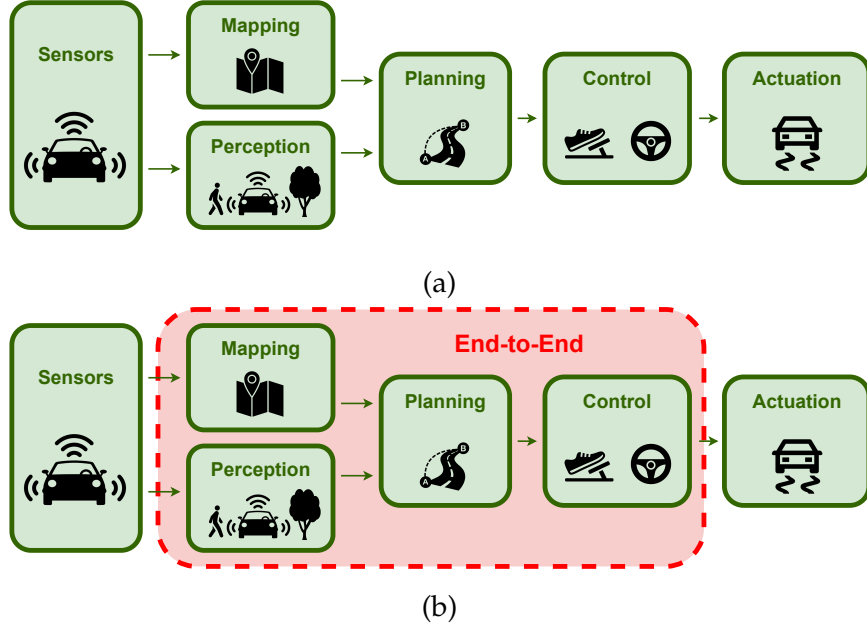


Figure 1.1: Architectures overview of an autonomous vehicle. (a) A generic modular architecture. (b) An end-to-end architecture.

- **Control.** This component translates the output of motion planning into actionable values for steering, acceleration, or braking. Essentially, it converts a high-level trajectory provided by the prior module into specific low-level values [335, 152].

The components of the modular pipeline can be developed utilizing deep learning techniques or classical non-learning methodologies (e.g. a learning-based object detector feeds information into a classical A-star path planning algorithm).

On the other hand, the end-to-end architecture combines all prior modules into one, directly deriving steering, throttle, and brake control values from sensor data. Due to its complexity, this approach solely relies on Deep Reinforcement Learning and Imitation Learning [298, 56, 131].

1.2 DEEP LEARNING FOR VEHICLE PERCEPTION

The advancements in self-driving technology are largely attributed to machine learning, particularly to deep learning (DL). Deep learning utilizes algorithms that mimic the structure and function of the human brain, known as artificial neural networks [100]. While the concept of artificial networks orig-

inated in the 1940s [209], its implementation feasibility and applicability have only been acknowledged in the last two decades. The increase in popularity is due to advancements in information and communication technologies, with the enhanced processing capabilities of graphical processing units (GPUs) and the vast availability of data, which have met the demanding computational and data needs of deep learning algorithms.

The precursor of neural networks is the perceptron model, a simple linear classifier proposed by Rosenblatt in 1957 [266]. When perceptrons are organized in multiple layers, they form a multi-layer perceptron (MLP) or an artificial neural network (ANN). Each layer processes the output from its preceding layer, allowing to extract features and learn meaningful data representations that correspond to different abstraction levels, from basic elements such as image edges and corners to more complex forms like shapes.

The theoretic expressive power of NNs, namely the type of functions they can implement, is considerable. Indeed, MLPs not only can implement any boolean function, but they are also universal function approximators. This means a feed-forward network with at least one hidden layer composed of a finite number of neurons can approximate continuous functions on compact subsets of $\mathbb{R}^d$, under mild assumptions on the activation function. However, the crucial problem is that the network dimension grows exponentially with the input size [282]. In order to understand the theoretic potentials of NNs, we can consider a geometric perspective: a single perceptron can express half-spaces, a network with one hidden layer can represent the intersection of half-spaces (i.e. convex sets), while a network with two hidden layers can express the union of intersections of half-spaces (i.e. non-convex sets).

Learning algorithms can be divided into three categories: supervised, unsupervised, and semi-supervised. In *supervised learning*, the model is trained using data that has predefined labels. In particular, the network processes the input data to infer output values, which are compared to the available labels using a loss function. The most common training method used is backpropagation [269], which computes the derivatives of the error with respect to weights and biases, from the output nodes back to the hidden layers, and updates the network parameters according to an optimization algorithm, such as stochastic gradient descent or Adam [268, 146, 198]. *Unsupervised learning*, on the other hand, does not use annotations. It aims to identify patterns or structures within the data, with common applications being clustering and association analysis.

1.2. DEEP LEARNING FOR VEHICLE PERCEPTION

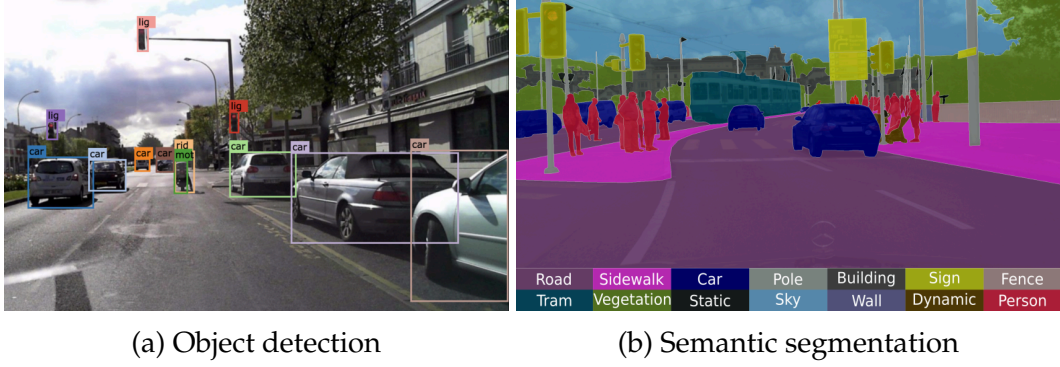


Figure 1.2: Examples of perception tasks. Sources: [354, 59].

Semi-supervised learning merges the two approaches, using both labeled and unlabeled data for training. This method can be beneficial when acquiring fully labeled data is challenging or costly.

The most prevalent deep learning architectures in contemporary research and application are Convolutional Neural Networks (CNNs) [165, 154] and Transformers [314]. Both architectures have revolutionized computer vision, natural language processing, and other fields, offering robust and efficient methods for a multitude of tasks. A comprehensive explanation of these architectures is provided in the appendix A.

1.2.1 PERCEPTION TASKS

Deep learning has been applied in several autonomous driving perception tasks. This section provides details about the most prevalent applications relevant to this study, including object detection, semantic and instance segmentation. While essential for self-driving vehicles, other tasks such as tracking [200, 52, 203, 242], depth estimation [351, 213], and multi-view 3D reconstruction [317, 229] are not addressed in this context.

Object Detection. Object detection, illustrated in Fig. 1.2a, is a fundamental requirement for achieving autonomous driving. Given that vehicles share the road with several other users, especially in cities, recognizing these obstacles is essential to prevent accidents. Detection in urban environments is hard due to the diverse object appearances and occlusions from other elements. Furthermore, similarities between objects or between an object and its surroundings, along with phenomena like shadows or reflections, add complexity to accurate

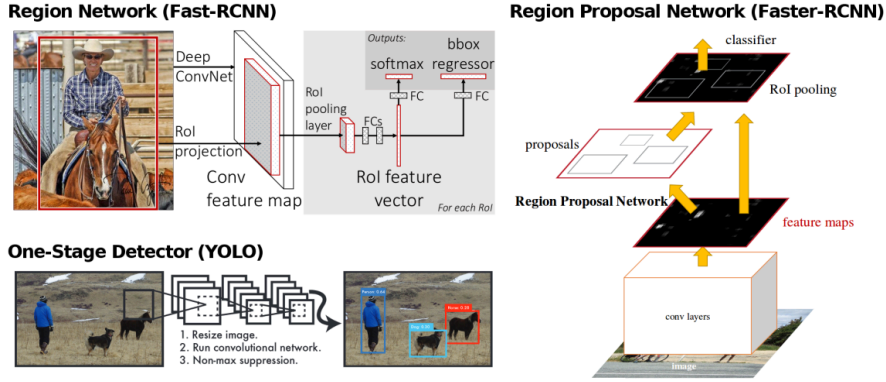


Figure 1.3: Illustration of three popular object detection networks. Sources: [130, 95, 261, 260].

detection. Classical object detection systems usually comprise a series of steps that are applied sequentially to solve the task. With the rise of deep learning, many of these steps [26, 84] and even the complete pipeline have been replaced by learning models [279, 94, 261, 260].

Sermanet et al. [280] introduced the use of CNNs for pedestrian detection by learning feature extraction in an unsupervised fashion using convolutional sparse auto-encoders. They employed a shallow network with a small receptive field for precise object localization using a sliding window. In comparison, deep networks with larger receptive fields made precise localization more challenging. Girshick et al. [94] introduced R-CNNs to address this localization issue, using a "recognition using regions" approach. They generated many region proposals with selective search [308], utilized a CNN for feature extraction from each proposal, and classified each region using a linear SVM. Despite being computationally intensive, several suggestions, such as He et al. spatial pyramid pooling [115] and Girshick Fast-RCNN [95], were made to reduce the computational load. Ren et al. [261] presented Faster-RCNN which incorporated Region Proposal Networks (RPNs) that shared convolutional features with the detection network, thus reducing additional computational costs.

One-stage detectors [279, 260, 191] later eliminated the region proposal step altogether, with [279] proposing the first one-stage detector that used a CNN-based sliding window approach. Redmon et al. [260] YOLO used a different technique, learning bounding boxes and class probabilities from feature maps of a network inspired by GoogLeNet [295], achieving real-time performance. Their subsequent version, YOLO9000 [259], even surpassed Region Proposal Networks

1.2. DEEP LEARNING FOR VEHICLE PERCEPTION

in performance. Liu et al. [191] enhanced both accuracy and efficiency by integrating feature maps from different scales and employing a fixed set of bounding boxes. Despite these advances, one-stage detectors still performed worse than region proposal algorithms, mainly due to the foreground-background class imbalance. Lin et al. [184] proposed a solution to this imbalance by introducing a dynamically scaled cross-entropy loss, which minimized the influence of easy examples. In contrast to earlier one-stage detectors that utilized anchor bounding boxes densely placed over images and refined using regression, Law and Deng [163] proposed to directly predict heatmaps for bounding box corners. Recently, models like DETR [34] have been introduced, leveraging the capabilities of transformers to address object detection challenges. Fig. 1.3 illustrates three popular object detection architectures.

For 3D object detection from 2D images, Zia et al. [385, 386] utilized 3D CAD models for enhanced object detail and pose estimations. Even though these 3D models provide detailed descriptions, they do not achieve comparable results to 2D bounding box detectors. To address this, Pepik et al. [241] combined 3D geometric representation with image matching. Further advancements [157, 155] integrated 3D data with 2D proposals.

Regarding 3D object detection from 3D point clouds, Li et al. [171] proposed a method that uses a 2D representation of this 3D data. Wang and Posner [318] adapted 2D detection techniques for 3D data, and subsequent methods like VoxelNet [380] optimized the process with end-to-end learning. However, due to the limitations of relying only on laser data, the integration of LiDAR and camera data has become an important line of research, showing potential for improved detection [49, 76]. Also as regards 3D detection, transformer-based architectures have been investigated to advance the field [215].

Semantic Segmentation. Semantic segmentation is a crucial task in computer vision, serving as an intermediate goal towards solving higher-level problems such as scene understanding or sensorimotor control. Semantic segmentation assigns a label from a predefined set of classes to each pixel in the image. The task is illustrated in Fig. 1.2b using an example from the Cityscapes dataset [59]. Segmentation of images into meaningful elements such as cars, pedestrians, or roads allows for a comprehensive understanding of the surroundings, which is fundamental to autonomous driving.

The success of deep convolutional neural networks in image classification

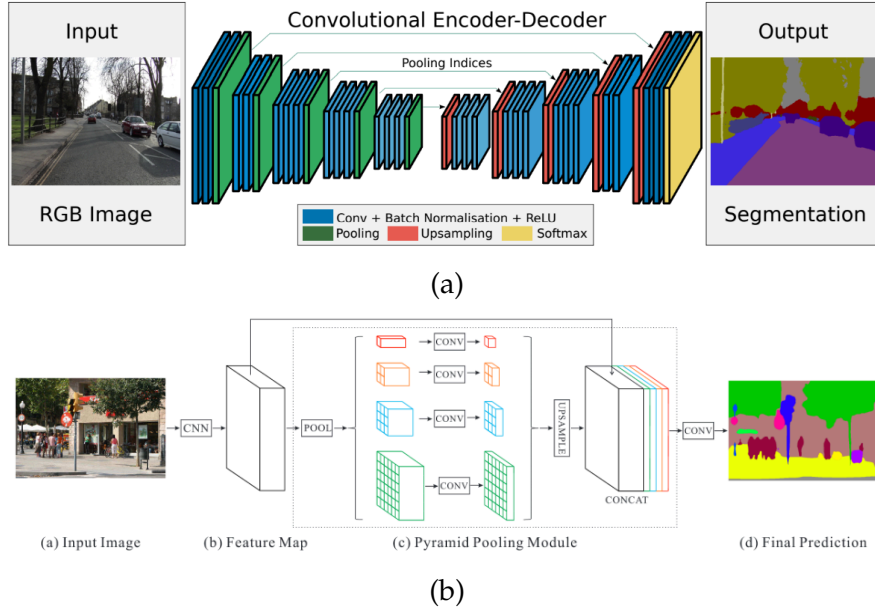


Figure 1.4: Architectures overview of semantic segmentation models. (a) Segnet. (b) PSPNet. Sources: [8, 373].

and object detection has led to their utilization in pixel-level tasks. The fully convolutional neural network [197] was an early application of CNNs for segmentation. Dilated convolutions [353] expanded the receptive field of these networks without increasing model size. Chen et al. [42] further built on this by using multiple dilated convolutions, proposing Atrous Spatial Pyramid Pooling (ASPP). On the other hand, in [8] an encoder-decoder approach with skip connections was introduced, which reduced the number of parameters required and enhanced segmentation boundaries (Fig. 1.4a). Ghiasi and Fowlkes [92] used a Laplacian pyramid based on CNNs to aggregate multi-scale information, refining boundaries. Pohlen et al. [248] introduced a ResNet-like architecture to maintain high-resolution details. Wu et al. [334] designed a more efficient ResNet model after observing their behavior as ensembles of shallow networks. PSPNet [372] better merges local and global contexts (Fig. 1.4b). Following this, [45] enhanced ASPP, showing results comparable to PSPNet. In [132] a method based on DenseNet [119] is introduced. Transformer-based models like SegFormer [336] have also emerged, further advancing the segmentation research. In conclusion, the problem of 3D semantic segmentation has been addressed using different input modalities, such as monocular image sequences [207], stereo image sequences [278] or 3D point clouds [87, 110].

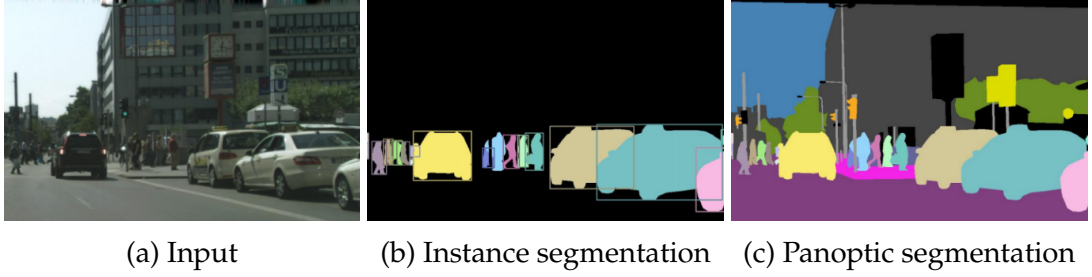


Figure 1.5: Instance and panoptic segmentation difference. Source: [150].

Instance and Panoptic Segmentation. The aim of semantic instance segmentation is to detect, segment and classify individual objects in an image. Contrary to semantic segmentation, this method provides details about the position, semantics, shape, and count of objects, thus it is widely applicable in the field of autonomous driving. Instance segmentation usually ignores classes unsuitable for the task, such as the sky or road. Conversely, panoptic segmentation, introduced by Kirillov et al. [150], focuses on comprehensively assigning both a semantic label and an instance identification, as depicted in Fig. 1.5.

Two primary research approaches exist for semantic instance segmentation: proposal-based and proposal-free methods. Proposal-based techniques typically involve two phases: extracting proposals and classifying them. On the other hand, proposal-free approaches directly predict pixel labels from the image itself. Proposal-based algorithms [244, 245] can be computationally slow due to the proposal generation phase. To alleviate this problem, Multi-task Network Cascade (MNC) [63] uses a fully convolutional network with three stages. Despite its efficiency, errors can occur from inaccurately scaled or shifted bounding boxes. Hayder et al. [112] introduced a shape-aware mask network integrated into the MNC framework to address this. While earlier methods tackled detection and segmentation sequentially, recent works like Mask R-CNN, FCIS and MaskLab [114, 178, 44] propose joint solutions. Proposal-based methods can inherit errors from the proposal generation phase, motivating the development of alternative techniques. Recent methods focus on jointly determining the segmentation and the semantic category of instances by treating instance segmentation as a pixel labeling task. Some strategies [367, 368] leverage depth information to discriminate object instances. However, other works [148, 6] propose proposal-free methods based on an initial semantic segmentation. For the panoptic segmentation task, several approaches have been proposed [150, 61, 149].

1.3 RESEARCH OBJECTIVES AND MOTIVATION

This section outlines the research objectives and the underlying motivation. Specifically, we address the concept of enhancing deep learning models and the methods employed for this purpose. Lastly, we highlight the contributions of our study to the existing literature.

1.3.1 WHAT DOES IT MEAN TO ENHANCE A DEEP LEARNING MODEL?

Enhancing a deep learning model generally means improving its results and performances on specific tasks. Given the generic and broad definition, this could manifest in several ways, depending on the particular problem the model is designed to address. The primary element to consider is the *metric*, which serves as a benchmark to evaluate the model performance, directly indicating its capability to solve the designated task. For example, in classification tasks, accuracy is the main metric, quantifying the proportion of correctly classified instances. A high accuracy indicates that the model has a superior capability to distinguish between categories. Similarly, for tasks like semantic segmentation, the Intersection over Union (IoU) measures the precision of the model pixel-level classification.

Metrics are not the only criterion to consider when evaluating a system. Another critical aspect is the model *generalization* capacity. Ideally, a well-generalizing method performs well not just on the training data, but also on new unseen data, avoiding the overfitting behavior. In terms of *robustness*, the objective is to make it more robust to adversarial attacks, noise, or other factors that could be potentially misleading. Furthermore, in contexts where models make decisions that impact people (e.g loan approvals or medical diagnoses), *fairness* is a fundamental principle. Improving fairness means reducing the model biases, ensuring no specific group is discriminated against. In addition, while *interpretability* does not directly influence the model performance, it is essential in helping to understand its decisions, which can lead to broader acceptance and more informed interventions if necessary. Finally, *resource efficiency* is another fundamental criterion to evaluate a deep learning system. Improving how efficiently it uses computational resources, like memory or processing power, can be important for deployment in constrained environments, like mobile or edge devices. In our study, we address model enhancement in terms of

metric and resource efficiency.

1.3.2 HOW TO ENHANCE A DEEP LEARNING MODEL?

Enhancing the performance of a deep learning model requires a systematic approach. Some widely adopted techniques include data-related approaches, methods based on the adaptation of the model architecture, and optimization and regularization algorithms:

- **Data-related approaches.** The success of a deep learning model largely depends on the data it is trained on. Proper data management involves not only the quantity but also the quality and diversity of the data. *Gathering more data* can expose the model to diverse scenarios, helping it identify more patterns [329]. Similarly, *generating more data* with other deep learning systems can help to enhance the model robustness, reducing the need for a time-consuming and costly data collection process [2]. Before training, refining the data through a *preprocessing* step is crucial [133]. This includes tasks like normalization, noise reduction, and managing missing values to prepare the data effectively. Preprocessing is also effective during the inference stage: high-quality input data can enable the model to detect subtle features that can make a significant difference in its performance. *Data augmentation*, where the dataset is artificially increased by applying random transformations, such as image rotation, to the original data, helps the model adapt to different inputs [286]. Equally important is addressing data imbalances to ensure the model does not favor any particular class over another.
- **Model architecture and complexity.** The architecture of a deep learning model influences its ability to extract meaningful features from the data. *Selecting the architecture* is fundamental to achieving the right balance in model complexity: a simple model with few parameters might underfit, while a complex model with too many weights risks overfitting [79]. *Transfer learning*, a technique that leverages pre-trained models on large datasets, enables models to be tailored for particular tasks [262]. *Knowledge distillation* is another approach where a smaller model learns to mimic a larger one, ensuring effective deployment with minimal loss in performance [102]. *Ensembles* enhance performance by integrating different

predictions, maximizing the benefits of each model [89].

- **Optimization and regularization.** They play important roles in training deep learning models efficiently and robustly. *Optimization* focuses on how the model learns and refines its weights. Adjusting the learning rate, choosing the right optimization algorithm (like SGD, Adam, or RMSprop), and setting their specific hyperparameters can significantly influence model convergence and overall performance [287]. *Regularization*, on the other hand, introduces constraints to reduce model overfitting [220]. Techniques such as dropout temporarily deactivate certain neurons during training, promoting more robust feature representations. L1 and L2 regularizations apply penalties to large weights in the model, preventing it from depending too heavily on particular data features. Fine-tuning hyperparameters for these strategies is a combination of expertise and experimentation to find the best settings for a given task.

In our research, we employ a variety of the previously mentioned techniques. Specifically, we generate synthetic data to boost detection and segmentation models, we enhance the input quality of 3D data to improve reconstruction, and we apply knowledge distillation to increase the computational efficiency of small models for classification, detection and segmentation tasks.

1.3.3 CONTRIBUTIONS

In our research, several contributions to the field are proposed. Firstly, in the context of generating synthetic training data, we introduce the concept of local domains, defining them as domain-specific spatial characteristics. Based on these priors, we propose a geometrical-guided patch-based image-to-image (i2i) translation approach, which enables a continuous geometrical translation. Through experiments in a few-shot setting, it is proven that our method enhances performance on all target proxy tasks, including lane detection and semantic segmentation [65]. Secondly, regarding the enhancement of input quality during inference, we present an innovative method for point cloud upsampling, allowing for arbitrary scaling factors with just one trained model. Instead of a straightforward upscaling, we propose learning a mapping from a low-resolution input point cloud to a probability distribution on a unit sphere. Sampling from this distribution and subsequently using the inverse

1.3. RESEARCH OBJECTIVES AND MOTIVATION

mapping results in a high-resolution output. This process is realized by a fully attention-based network architecture that demonstrates state-of-the-art performance across multiple benchmarks and showcases impressive generalization [64]. Lastly, we introduce a novel technique for feature-based Knowledge Distillation (KD) that enhances direct matching through a teacher features-driven regularization loss. By relaxing the feature matching constraint and distilling knowledge from a higher abstraction level, the student model can learn more robust and adaptable latent representations. The efficacy of the technique is validated on standard benchmarks, producing state-of-the-art results for various vision tasks, including image classification, both semantic and instance segmentation, and object detection [66].

From a broader perspective, we show how it is possible to leverage deep learning models to enhance other deep learning models. Through various techniques, we demonstrate that by using the knowledge and capabilities of one model, we can significantly boost the performance and efficiency of another. This strategy highlights the potential of collaborative learning, where models can work jointly to achieve better results.

2

Adversarial Networks for Synthetic Data Generation

The contributions of this chapter have been published in:

Dell'Eva, A., Pizzati, F., Bertozzi, M., and de Charette, R. (2022). Leveraging local domains for image-to-image translation. In International Conference on Computer Vision Theory and Applications (VISAPP). *Best paper award*.

2.1 INTRODUCTION TO GANs

Generative adversarial networks (GANs) are a revolutionary class of deep learning models introduced by Goodfellow et al. [101] in 2014. The framework has become a popular tool for several data synthesis and manipulation problems, especially in the visual domain, thanks to the distribution matching capability. It comprises two interacting neural networks: a generator G that aims to synthesize fake data that resembles real data, and a discriminator D that differentiates between real and fake data. In Fig. 2.1 the impressive progress of GANs on face synthesis over the past years is illustrated.

A successful training converges when the distribution of the produced data from G approximates the distribution of the original data. Indeed, GANs can learn complex high-dimensional distributions implicitly over images, audio, and other types of data. The discriminator serves as a guide for the generator, helping to refine the creations based on the current network weights, forming a

2.1. INTRODUCTION TO GANS



Figure 2.1: GAN progress on face synthesis. From left to right, results by the original GAN [101], DCGAN [256], CoGAN [189], PgGAN [143], StyleGAN [140], StyleGAN2 [142], and StyleGAN3 [141].

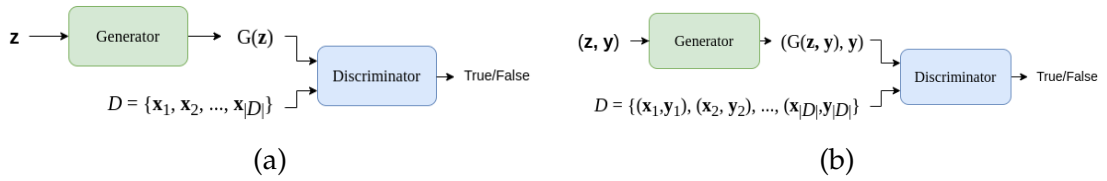


Figure 2.2: Unconditional vs conditional GANs. (a) In the unconditional setting, the noise input $\mathbf{z} \sim Z$ is mapped to a fake image $G(\mathbf{z})$. The discriminator differentiates real images $\mathbf{x}$ from the training dataset D from fake images from G . (b) The conditional setting includes an additional control signal $\mathbf{y}$, which could be another image (image-to-image translation), text (text-to-image synthesis), or a label (label-to-image synthesis).

dynamic loop of continual improvement in generating and discerning real and synthetic images. The framework is very flexible since the objective function for the generator is defined by the training dataset. For instance, if the training images contain *cars*, G must produce *cars* images to win the game. Similarly, if the training dataset includes *bicycle* samples, the generator output must contain *bicycle* to converge to a stable solution.

The GAN framework can be characterized by the capability to be conditioned by an auxiliary input. In the unconditional framework, as per the original GAN formulation [101], the generator is formulated as a mapping function from a simple distribution, such as a uniform or a Gaussian distribution, to a complex distribution, such as a natural image distribution. However, the lack of controllability in the output makes it inappropriate for many applications. In the conditional framework, instead, the goal of G is to generate outputs corresponding to the input control signal, which can take in different modes, including texts, images, and sounds. The two frameworks are compared in Fig. 2.2.

2.1.1 TRAINING AND EVALUATION

In the training process of GANs, a crucial step is the configuration of the objective function, which controls the dynamics between the generator and the discriminator to match the real data distribution p_{data} with the distribution of the generated data p_G , mapped from the simple latent distribution p_z . There are multiple metrics available to calculate the distance between the two distributions, including but not limited to, Jensen-Shannon divergence, Kullback-Leibler divergence, and integral probability metric. Due to the various ways to measure the distance between the distributions, several types of GAN losses have been formulated, including saturated GAN loss [101], non-saturated GAN loss [101], Wasserstein GAN loss [5], least-square GAN loss [206], hinge GAN loss [181], f-divergence GAN loss [136], and relativistic GAN loss [137]. The selection of the appropriate GAN loss is empirical and depends on the specific application.

The generator and the discriminator are trained jointly in a zero-sum game where G and D compete with one another. The parameters of D , denoted as θ_D , are optimized to maximize the loss function, aiming to accurately distinguish between real and fake samples. Conversely, the parameters of G , denoted as θ_G , are tuned to minimize the same loss, attempting to generate samples that D cannot differentiate from real data samples.

Formally, a generic GAN learning objective is formulated as:

$$\min_{\theta_G} \max_{\theta_D} V(G, D) = \mathbb{E}_{x \sim p_{data}} [\log D(x)] + \mathbb{E}_{z \sim p_z} [\log(1 - D(G(z)))] \quad (2.1)$$

In practice, the second term in Eq. 2.1 saturates, leading to a vanishing gradient problem and making G stop learning. To overcome the issue, the function is reformulated into two separate objectives:

$$\max_{\theta_D} \mathbb{E}_{x \sim p_{data}} [\log D(x)] + \mathbb{E}_{z \sim p_z} [\log(1 - D(G(z)))] \quad (2.2)$$

$$\min_{\theta_G} \mathbb{E}_{z \sim p_z} [\log D(G(z))]. \quad (2.3)$$

During the training process, two variants of stochastic gradient descent (SGD) schemes are commonly employed: the simultaneous update scheme and the alternating update scheme. In the simultaneous update, each training iteration comprehends both θ_G and θ_D update steps. This scheme allows for more efficient computation since a substantial part of the calculation in both steps can

2.1. INTRODUCTION TO GANS

be shared. On the contrary, in the alternating scheme, each iteration consists of one discriminator update followed by a generator update step. This tends to achieve a more stable training process as the generator update is informed by the latest state of the discriminator [24, 105, 190]. Notably, in [24, 105], the θ_D update is executed multiple times before proceeding with the θ_G update.

Regularization techniques are essential in stabilizing training and improving the quality of generated images. A common approach is Gradient Penalty (GP) [105, 267], which introduces an auxiliary loss term that penalizes any deviation in the gradient norm from a predetermined value. The term is added to the objective function for the discriminator update, i.e. Eq. 2.2, and effectively mitigates issues of vanishing and exploding gradients. Another notable technique is Spectral Normalization (SN) [216], which regularizes the spectral norm of each layer’s projection operation in the discriminator. This is achieved by adjusting the weight matrix with its largest eigenvalue, thereby ensuring that each projection layer maintains a spectral norm equal to one. Feature Matching (FM) [272] promotes the generator to produce images that resemble real ones by matching the moments of activations extracted from both real and fake images. A variant of FM that matches features extracted from real and fake images using a pre-trained network is named Perceptual Loss [135]. Finally, the Model Average (MA) technique is employed to enhance the quality of generated images by maintaining two versions of the generator network during training, i.e. the original G and the model average G . At each update step, a weighted mean between the two is computed.

Evaluating GAN models is challenging, with the most popular metrics being the Inception Score (IS) [272] and the Fréchet Inception Distance (FID) [117], both of which have significant limitations. The IS fails to detect intra-class mode collapse [23], allowing models that generate only a single image per class to score highly. Although FID addresses diversity measurement more effectively, it lacks an unbiased estimator. Another metric, Kernel Inception Distance (KID) [20], captures higher-order statistics with an unbiased estimator but suffers from high variance, potentially leading to inconsistent evaluations. A more recent metric, the Learned Perceptual Image Patch Similarity (LPIPS) [364], provides a method to determine the perceptual distance between two images. This metric, based on a weighted L2 difference between VGG16 embeddings [289], aligns well with human perceptions of image similarity, offering a promising alternative for GAN evaluation. Aside from these, there are metrics that separately assess

the fidelity and diversity of the generator distributions [159, 222]. Another widely used approach is manual inspection [68, 379], which allows for direct assessment of image quality. However, it has several limitations including being time-consuming, subjective, and constrained in assessing sample diversity. This method is often impractical during model development due to the substantial time requirement for reviewing numerous images. Additionally, it might not be reliable in certain domains, like the medical field, where distinguishing between realistic and non-realistic images demands specialized knowledge. Moreover, replicating results across different studies can be challenging due to variations in experimental protocols.

2.1.2 NETWORK ARCHITECTURE

Since the introduction of GANs, the architectures of both generator and discriminator networks have evolved to enhance the quality of the generated outputs. Initially, both employed multilayer perceptrons (MLPs), but G architecture advanced to adapt to the deep convolutional approach introduced in the DCGAN work [256], addressing the translation invariance of natural images that MLPs cannot handle efficiently. Further developments led to the integration of residual architectures [113], known to facilitate training with essential components like skip connections and activation normalizations (AN) (e.g. batch, instance, layer, and group normalizations [128, 309, 7, 332]). For conditional generators, several architectures have emerged, utilizing strategies like conditional AN, where the parameters are data-dependent [24, 125, 235]. As proved by several state-of-the-art generators [142, 321], another way to achieve controllability is to use hyper-networks, where auxiliary networks are employed to generate weights for the main network.

The evolution of discriminators, similarly, transitioned from MLP to deep convolutional and residual models. Effective architectures have been implemented to utilize control signals in D for enhanced image generation quality, encompassing auxiliary classifiers [227], input concatenation [129], and projection discriminators [217]. Recently, neural architecture search (NAS) has become popular in order to find a performant design for GANs [98].

2.1. INTRODUCTION TO GANS



Figure 2.3: (a) Example of super-resolution from [37], with the low-resolution input image on the left and the high-resolution output on the right. (b) Example of inpainting from [355]. From left to right: input, masked input, and inpainted result.

2.1.3 LIMITATIONS

GANs training presents substantial limitations that can hinder their development. The most well-known failure case is *mode collapse* [292, 10], where the generator fails to capture the diversity in the data distribution, producing examples from a limited number of modes. In the complete collapse case, G simply outputs a single example [4]. Additionally, GANs training suffers from *instability* and *non-convergence* problems, which arise from G and D oscillations during the process. Ideally, both networks should reach a fixed-point convergence (Nash equilibrium) where both are equally matched in terms of capabilities. However, the discriminator may easily differentiate between real and fake samples at the early stages if the generated samples are of low quality, causing the gradient to vanish and the generator to stop updating [101]. Other common issues include checkerboard and waterdrop artifacts. The proper selection of hyperparameters, such as batch size, learning rate, momentum, and weight decay, plays a crucial role in ensuring the training convergence [256].

2.1.4 APPLICATIONS

GANs have significantly influenced the development of many applications in the context of computer vision systems. Firstly, they have facilitated the evolution of image-to-image translation techniques, widely applied in image editing and domain adaptation [235, 53], as will be presented in the next section (Sec. 2.2). In our work, we leverage GANs to perform translations between domains where subtle local changes are crucial, e.g. highway and offroad. Secondly, GANs have been shown to produce higher quality results with respect to

traditional methods in tasks such as image restoration, which aims at recovering an original image that has been degraded by various factors (e.g. denoising and deblurring [40, 285]), image enhancement, that involves the manipulation of an image to make it more visually appealing (e.g. super-resolution [90, 37], as shown in Fig. 2.3a), and image inpainting, which fills missing or corrupted parts of an image with plausible data [355, 374, 376]. The images in Fig. 2.3b provide an example. Finally, they have exhibited great potential in the video synthesis domain, creating realistic human videos and transferring motions from one person to another, mainly for entertainment purposes [226], and in the neural rendering domain, augmenting the efficiency of the graphics rendering pipeline [223].

REMARK

Generative Adversarial Networks were extremely popular during the development of our work. Recently, another category of generative models, namely Denoising Diffusion Probabilistic Models (DDPMs) [70, 62], have gained more attention due to their outstanding results [257, 224]. They model the data generation process as a diffusion process, which is a series of reversible Markov chains. The process simulates the gradual corruption of data with noise, iterating backward to reconstruct the original data. Nevertheless, each framework has unique features, and recognizing the advantages and limitations inherent in each model is crucial for the specific use case.

2.2 IMAGE-TO-IMAGE TRANSLATION

The task of Image-to-Image Translation (i2i) aims at learning a mapping between images from a source domain and images from a target domain, transferring the main features while preserving the content. For example, the goal may be to convert a sketch into a photo-realistic image or change the season within a photo, as shown in Fig. 2.4. The problem can be tackled through supervised or unsupervised learning settings, depending on the availability of paired training data.

2.2. IMAGE-TO-IMAGE TRANSLATION

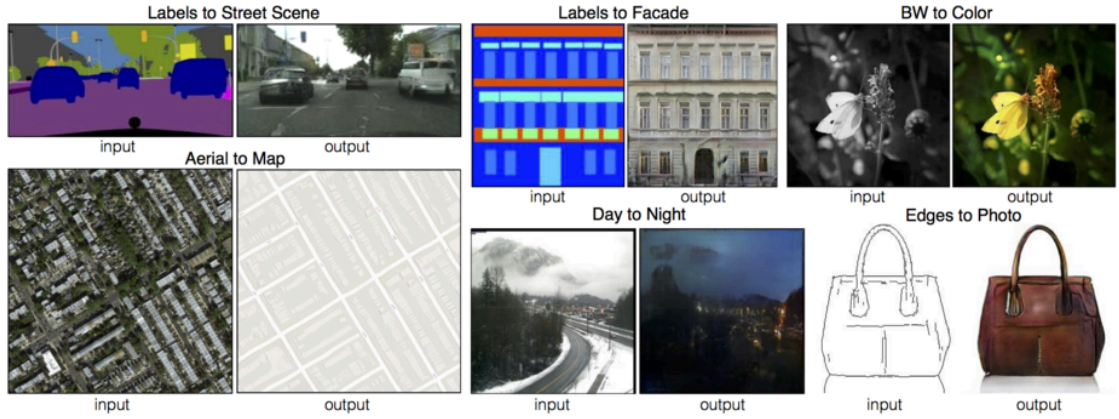


Figure 2.4: Supervised image-to-image translation examples from Pix2pix [129].

2.2.1 SUPERVISED IMAGE TRANSLATION

In the supervised setting, training pairs of corresponding images are available. Pix2pix [129] is a pioneering work that combines conditional GANs with pixel-wise L1 loss to ease general-purpose image-to-image translation. It employs PatchGAN discriminators that work on local patches instead of the entire image, alleviating the computational complexity of the model. Despite its effectiveness, one limitation of pix2pix is its one-to-one translation function, which prevents multimodal output distributions. This has led to developments such as Bicycle-GAN [381], which encourage output variations, and other works that enhance diversity and quality [166], including high-resolution image generation via pix2pixHD [322] and SPADE [235], which integrates spatially adaptive normalization layers for improved image quality and style control, as illustrated in Fig. 2.5.

2.2.2 UNSUPERVISED IMAGE TRANSLATION

Unsupervised image translation addresses the case where paired training data is unavailable, presenting a wider application range but increased complexity [14, 125, 145, 168, 382]. CycleGAN [382] introduces cycle consistency loss to ensure that an image, when translated to another domain and converted back, remains close to the original image. Various methodologies have been proposed, focusing on foreground object modifications without affecting the background [210] and on multimodal translation [125, 168]. MUNIT [125], for example, introduces disentangled latent space representations for multimodal

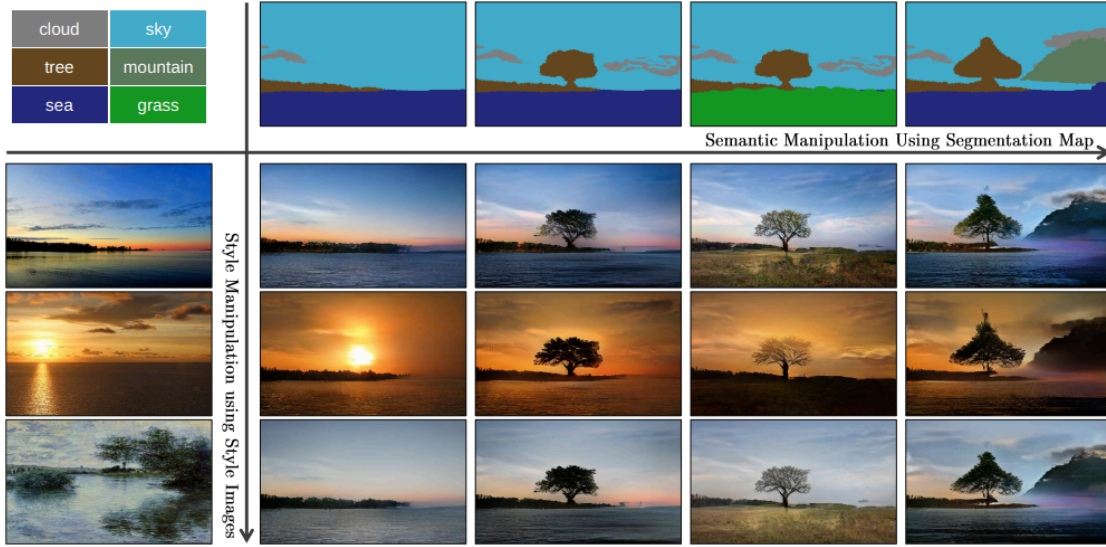


Figure 2.5: SPADE [235] allows user control over both semantics and style as synthesizing an image. The semantic is controlled via a label map, while the style is controlled via the reference style image.

outputs and example-guided translation.

Other works aim to reduce the dependency on a large number of training images, with techniques like FUNIT [190] and strategies for cases with limited images in either source or target domains [270]. Despite the advancements, unsupervised image-to-image translation remains an ill-posed problem, existing infinitely mappings that satisfy the constraints. Current approaches rely on the inductive bias of generator architectures to find meaningful mappings, assuming minimal complexity and changes to the input [88]. Recent works [138, 341] propose schemes that improve the quality, applicability and controllability of the existing translation models. Detailed insights into these developments can be found in the respective original papers.

2.3 STUDY OBJECTIVE AND CONTRIBUTIONS

Apart from their appealing translations, i2i GAN networks also offer an alternative to the supervised-learning paradigm. Indeed, as translations share features characteristics with the target domain they can be used to fine-tune proxy tasks, reducing the need for target annotations. However, i2i GANs perform well at learning global scene changes – winter $\rightarrow$ summer, paints $\rightarrow$ photos, etc. [188, 382], – but struggle to learn subtle local changes. Instead, we lever-

2.3. STUDY OBJECTIVE AND CONTRIBUTIONS

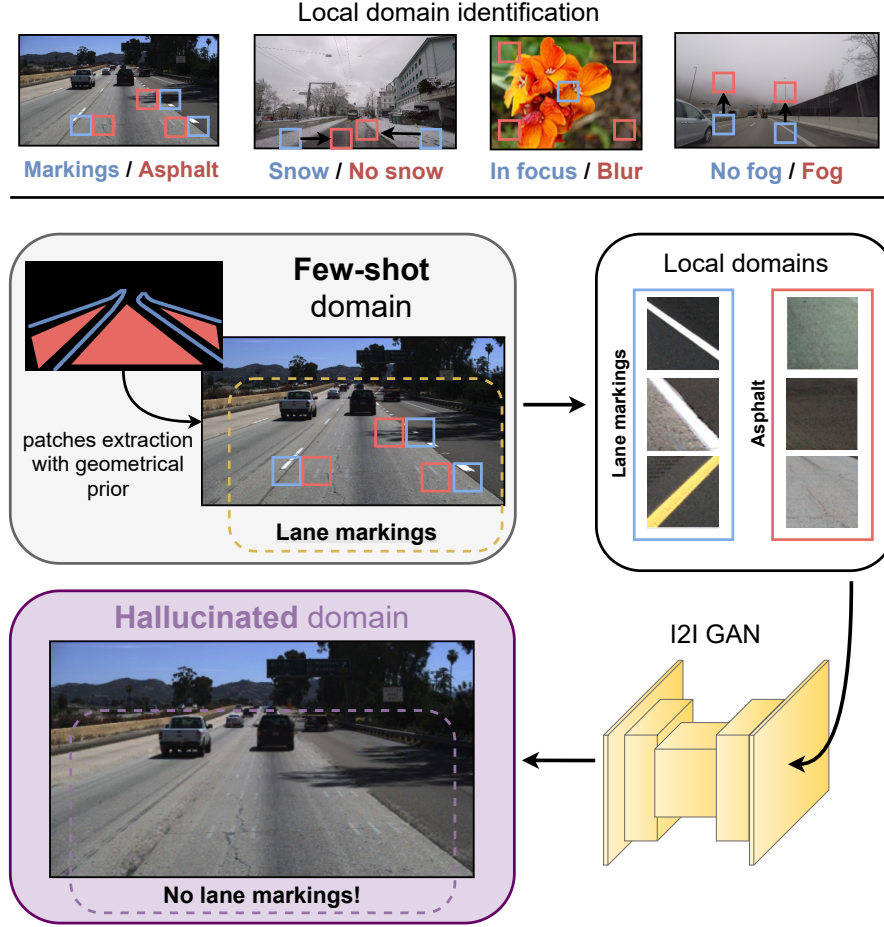


Figure 2.6: **Overview.** Our method is able to generate images of unseen domains, leveraging geometrical guidance to extract patches of *local domains*, i.e. spatially defined sub-domains, on source images. Here, we generate images without any lane markings training only on an extremely small amount of images with well-defined lane markings.

age human domain knowledge to guide i2i and improve proxy tasks on target, *without seeing target images*. This is of paramount importance for real-world applications like autonomous driving [275, 30, 264] which must operate safely in all hazardous conditions – some of which are rarely observed.

We propose a method exploiting human knowledge about source and target, to identify domain-specific local characteristics which we call *local domains* (Fig. 2.6, top). The latter are used as guidance to perform patches translations on *source only*, thus hallucinating a new unseen domain. An example in Fig. 2.6 bottom shows we leverage local domains knowledge about ‘lane markings’ and ‘asphalt’ to hallucinate a new domain without lane markings. Experimental evidence in this research shows that our new domain acts as a bridge leading to a

performance boost on target. An interesting characteristic of our method is that it trains few-shots on source only, leveraging only minimal human knowledge about the target.

In short, the main contributions of this study are:

- we introduce and define *local domains* as being domain-specific spatial characteristics (Sec. 2.5.1),
- to the best of our knowledge, we propose the first geometrical-guided patch-based i2i, leveraging our *local domains* priors (Sec. 2.5.2) and enabling continuous geometrical translation (Sec. 2.5.3),
- we experiment on three different tasks in a few-shot setting, showing that our translations lead to better performance on all target proxy tasks (Sec. 2.6).

2.4 RELATED WORKS

In this section, we present specific references relevant to our study. For broader references concerning i2i translation please refer to Sec. 2.2.

Image translation with less supervision. A recent field of study focuses on reducing the number of images necessary for training i2i networks. For instance, in BalaGAN [237] they exploit domain clustering in order to make the training robust to classes with few examples. Other strategies use self-supervision [325] or latent space interpolations [31] in order to avoid the discriminator overfitting and train on extremely small datasets. Differently, FUNIT [190] and COCO-FUNIT [270] generalize to few-shot domains at inference stage. Some other works try to work with less supervision at the domain level, on a mixed target domain [246] or without even source and target domain distinctions [9, 167]. It is worth noticing that some methods are trained on single images, as SinGAN [281], employable for image editing tasks. Finally, ZstGAN [183] exploits textual inputs for zero-shot image translation.

Prior-guided image translation. Several priors could be exploited to increase image translation effectiveness, with several degrees of supervision as bound-

2.4. RELATED WORKS

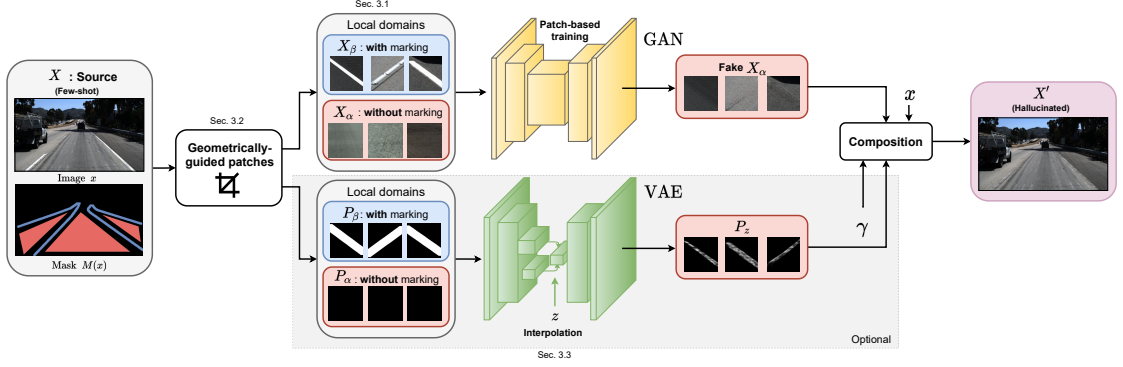


Figure 2.7: **Architecture pipeline.** Our method exploits knowledge about *local domains* (Sec. 2.5.1) and relies on geometrical-prior to extract samples of local domains in *source only* (Sec. 2.5.2) that train a patch-based GAN. Here, source has “lane markings” and “asphalt” local domains (X_α and X_β , respectively) while target has only “asphalt” (X_β), learning $X_\alpha \mapsto X_\beta$ further reduces the gap with target. An optional local domain interpolation strategy (Sec. 2.5.3) is added for generating geometrically continuous translation between local domains (here, simulating lane degradation).



Figure 2.8: **Translation with CycleGAN [382].** Sample output shows that i2i is prone to transfer global features (here, sky color) but neglects evident local features for humans as the street structure (note that IDD has no lane markings).

ing boxes [284, 19], semantic maps [174, 258, 301, 51, 384, 383, 182, 204, 235] or instance labels [217, 337]. Another line of works exploits physical models as priors for translation enhancement [111, 305], disentanglement [247], or guidance [246]. Importantly, scene geometry could be used as a prior, with learned correspondencies [331] or by exploiting additional modalities [3]. Some use text for image editing purposes [192]. Bruls et al. [30] exploit full semantic maps for road randomization, to generalize across challenging lane detection scenarios. However, they are limited to annotated road layouts and constrained by expensive complete segmentation maps.

2.5 METHOD

We address the problem of image-to-image translation accounting for *source* and *target* domains having predominant local transformations. As such, leveraging *only source* data, our proposal hallucinates a new *unseen* intermediate domain which can be used to ease transfer learning towards *target*. An overview of our pipeline is in Fig. 2.7.

In the following, we introduce our definition of local domains (Sec. 2.5.1) and propose a geometrical-guided patch-based strategy to learn translation between the latter (Sec. 2.5.2). For some local domains, we also show that a continuous geometrical translation can be learned from the interpolation of a mask (Sec. 2.5.3). Finally, we describe our training strategy showing few-shot capabilities (Sec. 2.5.4).

2.5.1 LOCAL DOMAINS

Image-to-image (i2i) networks learn a mapping function $G : X \mapsto Y$ from a source domain X to a target domain Y , such that the distribution $P_{G(X)}$ approximates P_Y . The goal is to transfer the features of domain Y to samples from X while preserving their content. This works well for transformation globally affecting the scene (eg. summer to winter) but struggles to capture the mappings of local changes due to the under-constrained settings of the system. A simple failure example, shown in Fig. 2.8, is the translation from outdoor images having lane markings, to images having no (or degraded) lane markings. As it seeks global changes, the i2i is likely to transfer unintended characteristics while missing the subtle – but consistent – local changes (here, the lane markings).

To overcome this, we introduce *local domains* which are sub-domains *spatially defined* – for example, lane markings, asphalt, etc. Formally, we define domain X as the composition of local domains, denoted $\{X_\alpha, \dots, X_\omega\}$, and the remaining sub-domains written X_O . Considering only two local domains of interest, it writes:

$$X = \{X_O, X_\alpha, X_\beta\}. \quad (2.4)$$

Because we consider only source and target domains sharing at least one local domain, say α , we write Y as:

$$Y = \{Y_O, Y_\alpha\}, \quad (2.5)$$

2.5. METHOD

so that the Kullback-Leibler divergence $\text{KL}(X_\alpha, Y_\alpha)$ is close to 0. Instead of learning the direct mapping of $X \mapsto Y$, we propose to learn local domain mappings, such as $X_\beta \mapsto X_\alpha$. If such mapping is applied systematically on all samples from X , we get a new domain X' without β , so:

$$X' = \{X_O, X_\alpha\}, \quad (2.6)$$

where domain X' is unseen and thus hallucinated. Considering that X' and Y share the same local domains, they are subsequently closer: $\text{KL}(Y, X') < \text{KL}(Y, X)$.

Our intuition is that when training target data is hard to get, our hallucinated domain X' can ease transfer learning. Notably here, our method only requires a priori knowledge of the shared local domains in source and target.

2.5.2 GEOMETRICALLY-GUIDED PATCHES

Learning the mapping between local domains requires extracting local domain samples. To do so we leverage patches corresponding to either local domains in the source dataset only. We rely here on a simple geometrical guidance from a mask $M(\cdot)$ to extract random patches centered around a given local domain.

Considering x an image in source domain X , we extract $\mathcal{X}_\alpha$ the unordered set of patches of fixed dimension, so that:

$$\mathcal{X}_\alpha = \{\{x_{p_0}, x_{p_1}, \dots, x_{p_m} \mid p \in M_\alpha(x)\} \mid \forall x \in X\}, \quad (2.7)$$

having m the number of patches per image, and $M_\alpha(x) = \llbracket M(x) = \alpha \rrbracket$ with $\llbracket \cdot \rrbracket$ the Iverson brackets. Literally, $M(x)$ is our geometrical prior – a 2D mask of same size than x – encoding the position of local domains. Subsequently, $M_\alpha(x)$ is filled with ones where local domain X_α is and zeros elsewhere. Similarly to Eq. 2.7, we extract the set X_β from $M_\beta(x)$ and X .

In practice, the geometrical prior $M(x)$ is often simply derivable from the image labels. For example, the position of lane marking and asphalt can both be extracted from image labels. In some cases, the position of local domains is constant dataset-wise and we use a fixed geometrical prior, so $M(x) = M$. This is for example the case for portraits datasets, where faces are likely to be centered and background located along the image edges.

Having collected the two sets of patches $\mathcal{X}_\alpha$ and $\mathcal{X}_\beta$, a straightforward patch-based GAN can learn $X_\alpha \mapsto X_\beta$. In some cases, $\mathcal{X}_\alpha$ and $\mathcal{X}_\beta$ being of similar nature we demonstrate spatial interpolation is beneficial.

2.5.3 LOCAL DOMAINS INTERPOLATION

Continuous i2i are extensively studied [97, 324, 160], but existing methods are not suitable for translation affecting only local regions as in our problem setting (see Sec. 2.6). Instead, we learn a non-linear geometrical interpolation of patch masks, leveraging a variational autoencoder (VAE).

Previously we described each patch as encompassing a single local domain but, in reality, patches often mix multiple local domains. This is the case of lane markings patches, shown in Fig. 2.7, that contain asphalt too. Hence, along with the set of local domains patches we extract the sets $\mathcal{P}_\alpha$ and $\mathcal{P}_\beta$ directly from our geometrical guidance $M(\cdot)$, and seek to continuously interpolate $P_\alpha \mapsto P_\beta$.

In practice, our VAE having encoder $E(\cdot)$ and decoder $D(\cdot)$ is trained in the standard fashion, but at inference it yields the latent representation h_Z corresponding to the linear combination of $E(p_\alpha)$ and $E(p_\beta)$, having $p_\alpha \in \mathcal{P}_\alpha$ and $p_\beta \in \mathcal{P}_\beta$, respectively¹. Formally:

$$\begin{aligned} h_Z &= E(p_\alpha)z + E(p_\beta)(1 - z), \\ p_z &= D(h_Z), \end{aligned} \tag{2.8}$$

where $z \in [0, 1]$ encodes the progress along $P_\alpha \mapsto P_\beta$. The final interpolated patch x_z is the composite between x_α and x_β patches, following the VAE output. It writes:

$$\begin{aligned} x_z &= x_\alpha m + x_\beta (1 - m), \\ \text{with } m &= \gamma p_z, \end{aligned} \tag{2.9}$$

$\gamma \in [0, 1]$ being an arbitrary controlled blending parameter adding a degree of freedom to our model. Furthermore, notice that the stochastic VAE behavior further increases variability, beneficial for proxy tasks.

¹In our formalism, we include the VAE reparametrization in $E(\cdot)$

2.6. EXPERIMENTS

2.5.4 TRAINING

We train our pipeline, the patch-based GAN and the optional VAE, leveraging only images from the source domain and geometrical priors about local domains. The patch-based GAN is trained on $X_\alpha \mapsto X_\beta$ (Sec. 2.5.2) minimizing the LSGAN [206] adversarial loss:

$$\begin{aligned} y_f &= G(x), \\ \mathcal{L}_G(y_f) &= \mathbb{E}_{x \sim P_X(x)} [(D(y_f) - 1)^2], \\ \mathcal{L}_D(y_f, y) &= \mathbb{E}_{x \sim P_X(x)} [(D(y_f))^2] + \\ &\quad + \mathbb{E}_{y \sim P_Y(y)} [(D(y) - 1)^2], \end{aligned} \tag{2.10}$$

along with task-specific losses. If used, the VAE interpolation (Sec. 2.5.3) is trained with standard ELBO strategy [21], minimizing reconstruction loss along with a regularizer:

$$\begin{aligned} \mathcal{L}_{VAE} &= -\mathbb{E}_{q_\phi(z|x)} \log p_\theta(x|z) + \\ &\quad + D_{KL}(q_\phi(z|x) || p(z)). \end{aligned} \tag{2.11}$$

At inference time, the full image is fed to the GAN backbone to produce the translated image, while the corresponding full interpolation mask is obtained processing mask patches independently and then stitching them together with a simple algorithm. Of note, our method has important few-shot capabilities. As we train only on source patches a reduced number of image samples is sufficient to get reasonable data diversity, which we further demonstrated in the following section.

2.6 EXPERIMENTS

We evaluate our method on 3 different tasks, namely lane markings degradation, snow addition and deblurring, leveraging 5 recent datasets [307, 313, 271, 60, 225], and evaluating our translation both against i2i baselines and on proxy tasks. In Sec. 2.6.1 we provide details on our tasks, while Secs. 2.6.2, 2.6.3 report extensive qualitative and quantitative evaluation.

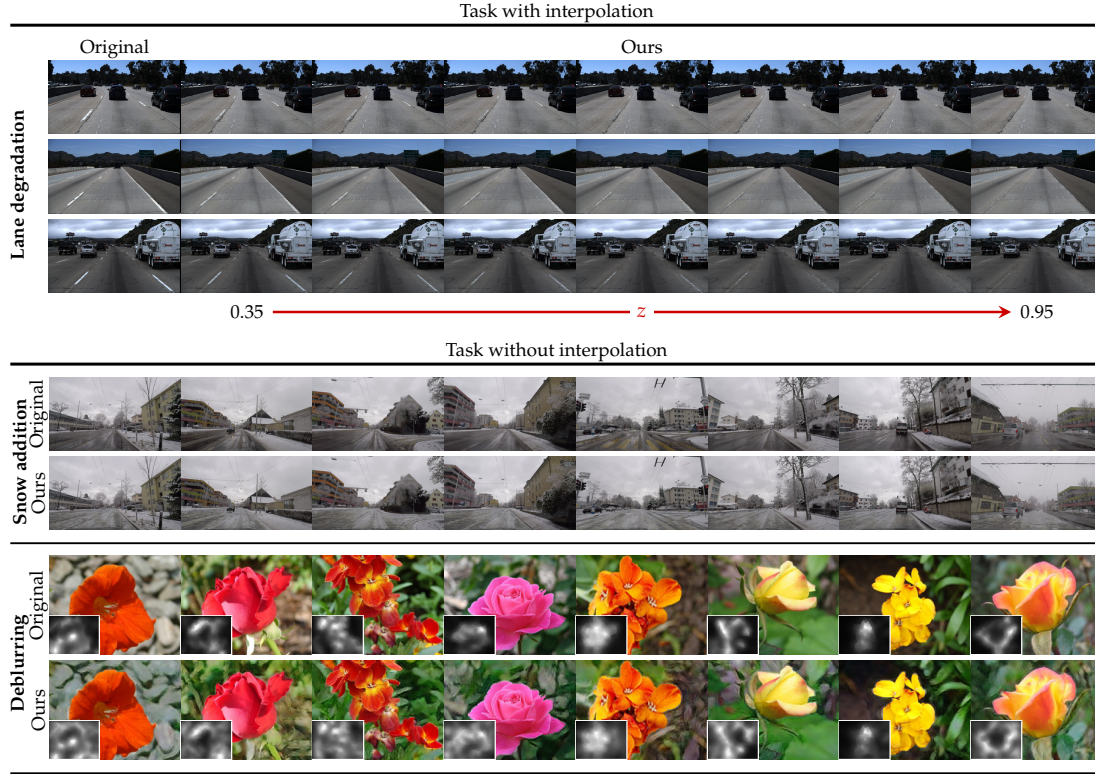


Figure 2.9: **Qualitative results.** For each task we show the original input image and our output with the $X_\beta \mapsto X_\alpha$ local domains translation. **Lane degradation:** sample translations on TuSimple [307] test set with increasing degradation $z \in [0.35, 0.95]$ from left to right, blending variable $\gamma = 0.75$. **Snow addition:** augmentation of ACDC [271] validation set, only road is involved in the transformation. **Deblurring:** Original flower images [225] and our deblurred version. Bottom left insets show the in-focus map [96] which are whiter on average (i.e. less blurred) for ours.

2.6. EXPERIMENTS

2.6.1 TASKS DEFINITIONS

We describe our three task below, detailing the learned local domains translation $X_\beta \mapsto X_\alpha$.

Lane degradation. Here, we leverage the highway TuSimple [307] dataset having clear lane markings. For local domains, we chose *lane marking* (X_β) and *asphalt* (X_α) exploiting geometrical priors from the provided lane labels and assuming near by asphalt. We use our interpolation strategy (Sec. 2.5.3) accounting for both degradation and blending. Importantly, we train only on 15 images (1280x720) to demonstrate few-shot capabilities, with 30 patches per image of size 128x128, 200x200 and 256x256. Backbones are DeepFillv2 [355] as GAN and IntroVAE [120] for interpolation. The latter is trained with a binarized difference mask from lane inpainting and original image.

We evaluate our translations on the standard 358/2782 val/test sets of TuSimple. In addition to demonstrating generalization, we evaluate several lane detectors on 110 images from the India Driving Dataset (IDD) [313] – never seen during training – having degraded lane markings which we manually annotated.

Snow addition. Here, we rely on snowy images from the recent Adverse Driving Conditions Dataset (ACDC) [271], which typically have snow only on sidewalk and *not* on road. The task is to add snow on the road. Logically, local domains are *road* (X_β) and snowy *sidewalk* (X_α), exploiting semantic labels as priors. Again, we train only with 15 images with 30 patches (128x128) per image, using CycleGAN [382] with default hyperparameters. No interpolation is used. We evaluate on the original val/test set of ACDC having 100/500 images. To increase generalization for the segmentation task in snowy weather, we also augment Cityscapes [60] with the same trained network.

Deblurring. We leverage the Oxford 102 Flower dataset [225] to learn turning shallow Depth of Field (DoF) photos to deep DoF, therefore seeking to deblur the image. As blur is not labeled, we rely on a simple *dataset-wise* geometrical prior that image center is always in-focus and image corners are always out-of-focus. Local domains are *out-of-focus* (X_β) and *in-focus* (X_α). Since we use only 8 patches per image (4 in-focus, 4 out-of-focus, 128x128), we train our CycleGAN [382] with 400 images, adding a task-specific objective function defined as the composition

of a color consistency loss and an in-focus loss:

$$\mathcal{L}_{deblur} = D_{KL}(H[x]||H[G(x)]) + \frac{1}{\sigma_{LoG(G(x))}^2}, \quad (2.12)$$

with $H[.]$ the image histogram and $\sigma_{LoG(.)}^2$ the Laplacian of Gaussian variance. A color jitter augmentation is applied to ensure discriminator invariance to color. We do not use interpolation. At inference, we exclude foreground since it impacts translation quality due to the identity loss in CycleGAN [382].

Images	In-focus avg↑
Original	1.28
Ours (<i>deblurred</i>)	1.53

Table 2.1: **Deblurring performance.** Average of the in-focus maps [96] on the Oxford Flowers [225] test set show our method efficiently deblur the input images despite a trivial dataset-wise geometrical prior.

2.6.2 EVALUATION

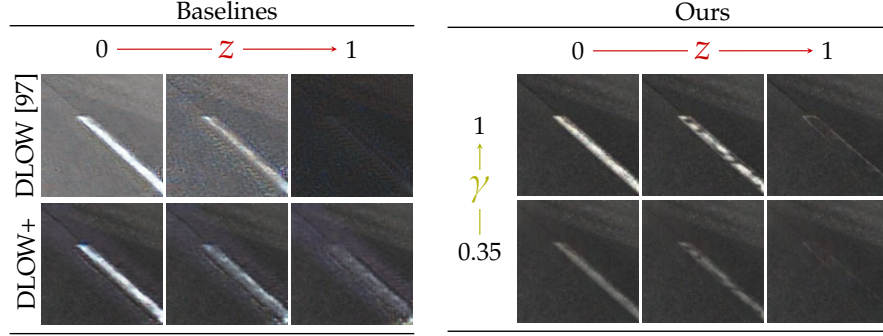
TRANSLATION QUALITY

Qualitative results are visible in Fig. 2.9 and show our method outputs realistic translations for all tasks. In details, we are able to modify lanes (first three rows) on TuSimple with different degrees of degradation (from left to right). On snow addition, images show plausible snow on ACDC roads (middle two rows), preserving shadows.

Finally, on our deblurring task (bottom two rows) the flowers background appear in-focus, exhibiting sharper edges. To better size the benefit on this last task, flower images have as inset the in-focus map computed with [96] (ie. white means in-focus).

Of note, evaluating GAN metrics on target images would be biased since we use only source images, unlike existing i2i. They are reported in Sec. 2.7.1 for the sake of completeness. To provide a quantitative quality evaluation, Tab. 2.1 reports the in-focus average proving our translations are significantly more in-focus (+0.24) than original images.

2.6. EXPERIMENTS



(a) Qualitative

Network	FID↓	LPIPS↓
DLOW [97]	211.7	0.4942
DLOW+	155.6	0.4206
Ours w/o blending	154.7	0.3434
Ours	135.4	0.3254

(b) GAN metrics

Figure 2.10: **Lane translations.** (a) Qualitative comparison of lane degradation on patches with baselines. Our method is the only one to output a realistic degradation. (b) GAN metrics on the lane degradation task prove the benefit of our method.

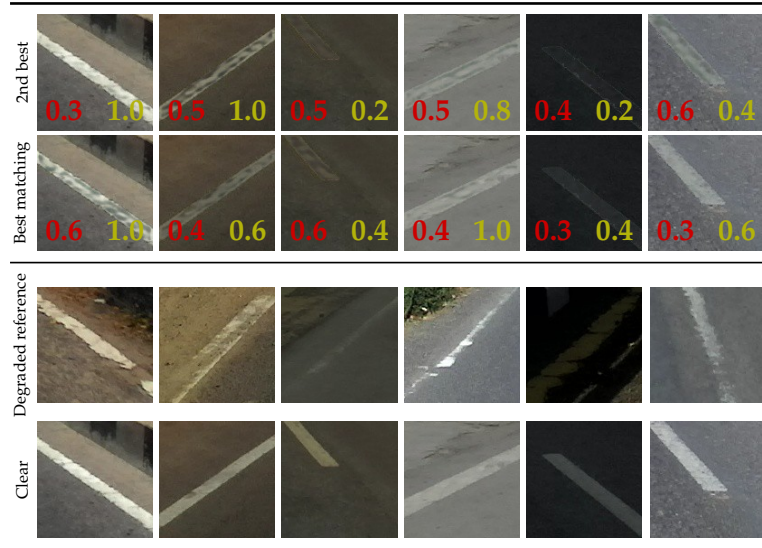


Figure 2.11: **Evaluation of lane degradation on patches taken from IDD dataset [313].** We associate clear patches (bottom row) to degraded ones (third row) by minimizing LPIPS. Applying our method to clean images varying the z and γ parameters (shown in the images), we subsequently lower the LPIPS. We display the best and second best translation in terms of LPIPS. The similarities of our results with the degraded patches prove the efficacy of our LPIPS-based evaluation.

INTERPOLATION QUALITY

For the lane degradation task, we compare our interpolations against the continuous i2i DLOW [97] baseline, trained on the same data. As it suffers from evident color artifacts, we introduce DLOW+: a custom version using lane mask as additional channel input, masked reconstruction loss, and masked input-output blending. For DLOW/DLOW+, we regulate the walk on the discovered manifold of each network with a *domainness* variable z – which amounts to our lane degradation. With respect to baselines, it is visible in Fig. 2.10a that our degraded lane translations are more realistic for all z since DLOW and DLOW+ discover simpler transformations, just regulating color homogeneously.

For quantification, we compare translations against real degraded lane markings from IDD and report FID and LPIPS in Fig. 2.10b. In details, we select 35/62 clear/degraded lane patches from IDD test set, and couple those with minimum LPIPS [364] distance. Intuitively, we pair similar clear and degraded lane markings together. Pairs are shown in the two bottom rows of Fig. 2.11. We then degrade each clear image with ours / DLOW / DLOW+, generating several degraded versions, and use the best degrading version in terms of LPIPS w.r.t. its clear match to compute GAN metrics. Fig. 2.10b shows we outperform baseline on both metrics significantly (roughly, -20 FID, and -0.1 LPIPS), demonstrating the realism of our lane degradation. Since baselines are not using any explicit blending as us (see γ in Eq. 2.9), we also evaluate “ours w/o blending” using $m = p_z$ in Eq. 2.9, which still outperforms baselines.

PROXY TASKS

Here, we study the applicability of our pipeline to increase the robustness of existing lane detection and semantic segmentation networks.

Lane detection. We aim here to make lane detectors robust to *unseen* degraded lane markings. To do so, we train two state-of-the-art detectors (SCNN [230] and RESA [377]) on both TuSimple original images and our translated version (mixing with 5% probability and randomizing z and γ). The models are tested on both the TuSimple test set and our 110 labeled IDD images, the latter having severely degraded lane markings.

From the quantitative results in Tab. 2.2, we observe that with our source degraded translations both detectors severely outperform the baselines using

2.6. EXPERIMENTS

Detector	Translation	TuSimple [307]			IDD [313]		
		Acc. $\uparrow$	FP $\downarrow$	FN $\downarrow$	Acc. $\uparrow$	FP $\downarrow$	FN $\downarrow$
SCNN [230]	None (source)	0.946	0.052	0.069	0.617	0.538	0.741
	Ours	0.945	0.058	0.072	0.730	0.453	0.577
RESA [377]	None (source)	0.952	0.056	0.065	0.639	0.720	0.800
	Ours	0.951	0.059	0.068	0.671	0.686	0.761

Table 2.2: **Lane detection on TuSimple and IDD.** Performance of lane detectors when trained on TuSimple source (*none*) or our degraded translations (*ours*). The latter significantly outperforms baseline, while retaining equivalent performances on TuSimple images.



Figure 2.12: **SCNN [230] lane detection on IDD [313].** Training on generated images with degraded lanes makes existing lane detectors (e.g. SCNN [230]) resistant to scenes with damaged (first three columns) or no (last column) lane markings.



Figure 2.13: **Cityscapes images with snow added.** We add snow on roads and sidewalks of the Cityscapes training set to train semantic segmentation networks robust to snow. Cityscapes exhibits a domain shift with respect to ACDC, but our method is still able to generate acceptable snow.

Model	Translations	road IoU $\uparrow$	sidewalk IoU $\uparrow$	mIoU $\uparrow$
DeepLabv3+ [43]	None (source)	74.95	39.52	45.31
	Ours	80.56	49.52	47.64
PSANet [371]	None (source)	74.29	30.71	42.97
	Ours	74.01	36.28	43.85
OCRNet [360]	None (source)	82.30	45.60	54.54
	Ours	82.78	54.69	55.48

Table 2.3: **Semantic segmentation on ACDC [271] snow.** We train multiple segmentation network on Cityscapes [58] with added snow with our method and test on ACDC [271] snow validation, consistently improving generalization capabilities.

clear source on the challenging IDD, while maintaining on-par performances on TuSimple with clear markings. In particular, for SCNN we improve by +11.3% the accuracy, -8.5% the false positives and -16.4% the false negatives. Sample qualitative results are in Fig. 2.12 and showcase the robustness of our method on degraded or even absent street lines. We conjecture that our degraded translations forced the network to rely on stronger contextual information.

Semantic segmentation. Here, we seek to improve segmentation in snowy driving conditions. We train three state-of-the-art semantic segmentation models, namely DeepLabv3+ [43], PSANet [371] and OCRNet [360], with either clear Cityscapes images and snowy Cityscapes images translated with our method. Translation examples are available in Fig. 2.13, where we add snow on sidewalks and road by using Cityscapes semantic maps. Visual results remain acceptable and snow is added uniformly on both semantic classes, even if inference on Cityscapes brings a consistent domain shift with respect to training patches

2.6. EXPERIMENTS

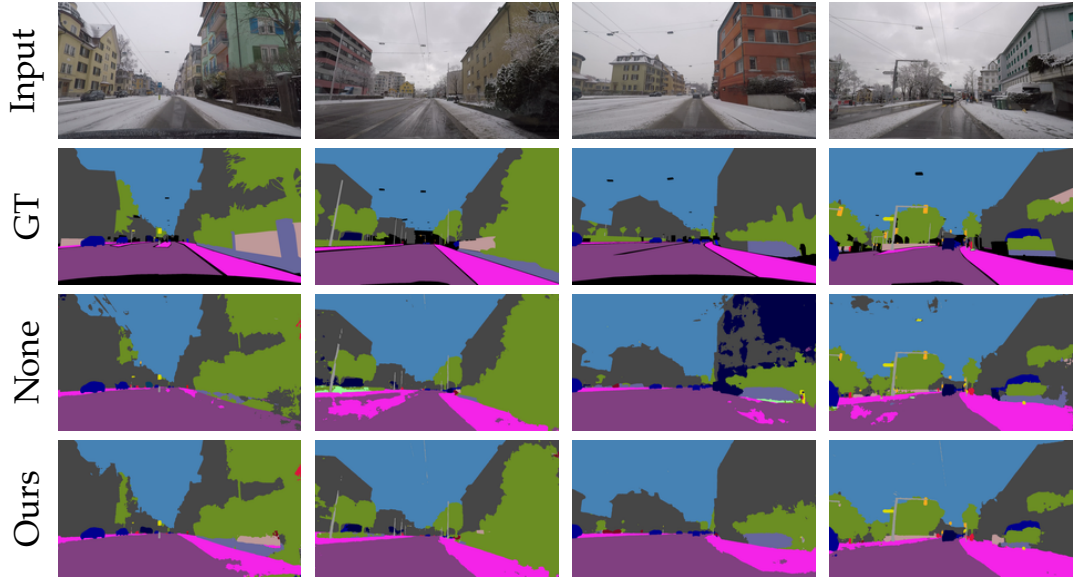


Figure 2.14: **DeepLabv3+ [43] on ACDC [271] snow.** Training with our generated images brings improvements in segmentation of snowy scenes in ACDC [271], especially in the road and sidewalk classes.

on ACDC. In detail, for the latter we augment images with 10% (DeepLabv3+, PSANet) or 5% (OCRNet) probability. The models are evaluated on the ACDC snow validation set.

Tab. 2.3 shows the benefit of our augmented images (*Ours*) to consistently improve the performance on road or sidewalk (our two local domains) and mean IoU for all networks. From Fig. 2.14 it is visible that the model trained with our augmentation strategy is able to better detect roads and footpaths in difficult weather conditions with respect to the baseline, which is not capable of properly discriminating between them if they are covered with snow.

2.6.3 ABLATION STUDY

Training images and patches. As mentioned our method requires very little images to train. Here, we study the effect of number of images and patches per image on the lane degradation task. To measure its impact, we use LPIPS [364] and FID [117] following Sec. 2.6.2.

Results in Tab. 2.4 show, as expected, better translation with the increase of both the number of images and the number of patches extracted per each image. However, we also denote the few-shot capability of our method, and the minimal benefit of using a large number of images.

TuSimple samples (%)	Patches/img	LPIPS↓	FID↓
15 (0.4%)	1	0.3296	148.22
15 (0.4%)	5	0.3295	135.53
15 (0.4%)	30	0.3254	131.73
15 (0.4%)	60	0.3246	127.94
15 (0.4%)	150	0.3222	126.94
50 (1.4%)	30	0.3236	129.42
150 (4.1%)	30	0.3221	124.79
500 (14%)	30	0.3234	128.56
3626 (100%)	30	0.3218	125.56

Table 2.4: **Data ablation on TuSimple.** The use of data on the lane degradation task (TuSimple $\mapsto$ IDD) is ablated by varying the number of images and patches per images in the training set, and evaluating GAN similarity metrics (see Sec. 2.6.2) on IDD.

Augmented images (%)	road IoU↑	sidewalk IoU↑	mIoU↑
100	36.43	35.26	39.83
66	60.43	43.81	46.84
50	68.78	45.79	50.31
20	75.85	44.60	47.36
10	80.56	49.52	47.64
0 (none)	74.95	39.52	45.31

Table 2.5: **Augmentation percentage ablation on Cityscapes.** The effectiveness of our snow addition translation is ablated by varying the probability of Cityscapes augmented images shown to DeepLabv3+ [43] during training. Segmentation evaluation is reported on ACDC [271] validation set.

2.7. SUPPLEMENTARY RESULTS

Augmentation percentage. We study also how the percentage of augmented images shown to DeepLabv3+ network at training impacts performances on semantic segmentation in snowy conditions.

As indicated in Tab. 2.5, we achieve best performances with an augmentation probability of 50% (+5% mIoU w.r.t. no augmentation), still we use for evaluation in Sec. 2.6.2 the model obtained with 10% for its higher accuracy on road and sidewalk – crucial for autonomous navigation tasks.

2.7 SUPPLEMENTARY RESULTS

We provide additional results of our method. In Sec. 2.7.1, we propose an additional comparison with baselines for the snow addition task. Also, in Sec. 2.7.2, we display additional qualitative results for image-to-image translation and the proposed proxy tasks.

2.7.1 GAN METRICS

For completeness, we report GAN metrics against CycleGAN [382] and CycleGAN-15, which are trained on $\text{ACDC}_{\text{clear}} \mapsto \text{ACDC}_{\text{snow}}$ using respectively 400/400 or 400/15 source/target images. Comparatively, we *only* use the same 15 cherry-picked $\text{ACDC}_{\text{snow}}$ images. The quantitative evaluation is obtained by performing roads and sidewalks translation of 100 $\text{ACDC}_{\text{clear}}$ images relying on segmentation masks from OCRNet [360] pretrained on Cityscapes [60]. Since ACDC provides images (weakly) paired, we compute the pair-wise average LPIPS metric between each fake translation and its paired real snow image. We also evaluate FID between the fake and real snow datasets. It is important to note that we do not seek to outperform the baselines since they have access to $\text{ACDC}_{\text{clear}}$ images while our method does not. Results in Tab. 2.6 however show we perform reasonably good given the additional domain gap, even on par with baselines on LPIPS metric. In addition, in Fig. 2.15 our translations are shown to be significantly more homogeneous than baselines.

Network	Training samples		FID↓	LPIPS↓
	clear	snow		
CycleGAN [382]	400	400	110.30	0.6225
CycleGAN-15	400	15	111.29	0.6271
Ours	0	15	123.14	0.6283

Table 2.6: **Snow translation similarity on $\text{ACDC}_{\text{snow}}$.** GAN metrics on the snow addition task confirm the validity of our model. Without using any target images, our model yields acceptable results on FID, attaining even almost on-par performances with baselines on LPIPS.



Figure 2.15: **Qualitative snow addition comparison on $\text{ACDC}_{\text{clear}}$.** Our method (last row) yields acceptable performances compared with different strategies (rows 2-3). Please note that we are the only one to *not* use $\text{ACDC}_{\text{clear}}$ images for training.

2.7. SUPPLEMENTARY RESULTS

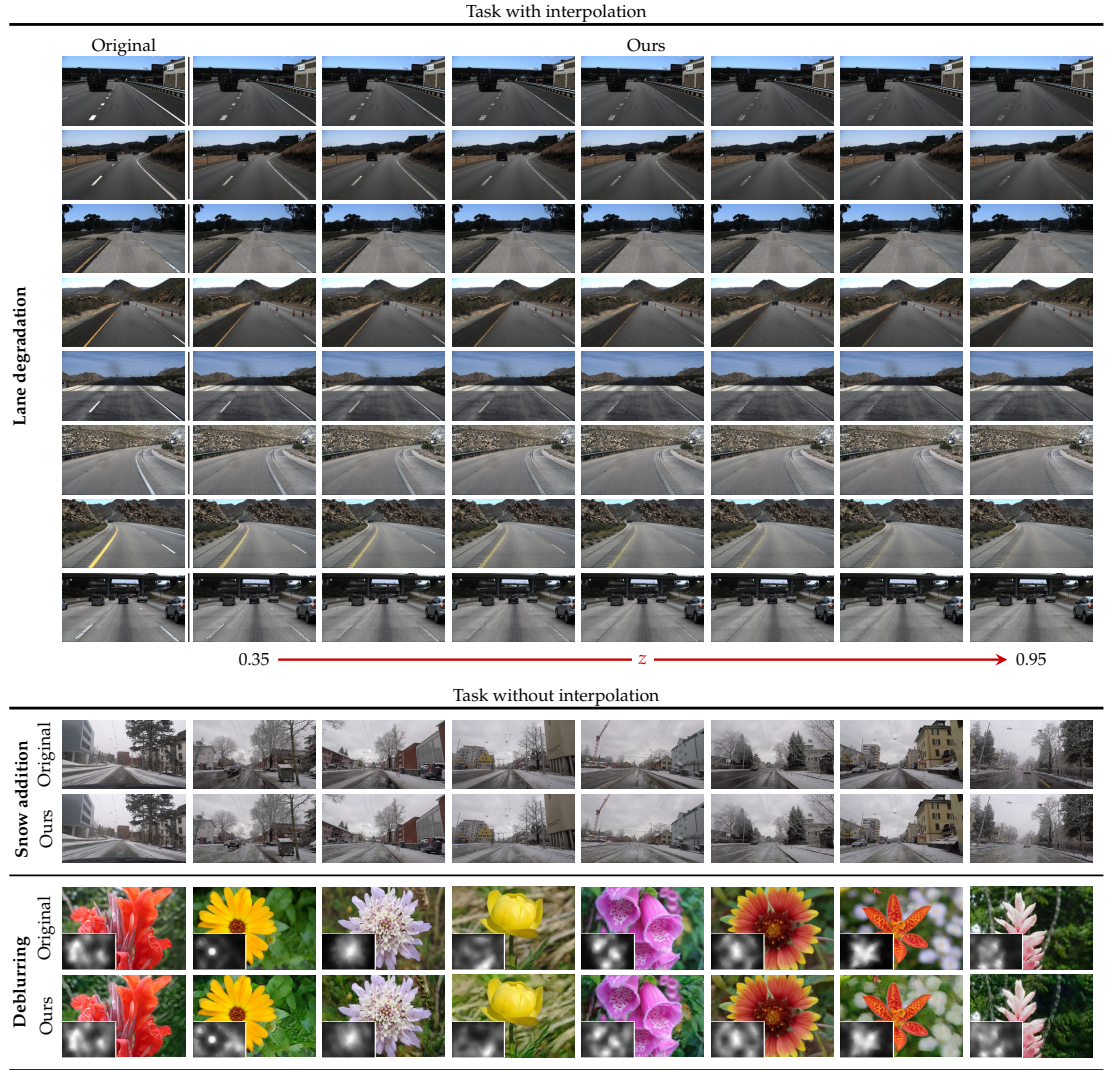


Figure 2.16: **Qualitative results.** Additional qualitative results for the lane degradation, snow addition and deblurring tasks.

2.7.2 ADDITIONAL QUALITATIVE RESULTS

IMAGE-TO-IMAGE TRANSLATION

Additional i2i qualitative results are shown in Fig. 2.16. With our pipeline, we are able to realistically modify TuSimple [307] street lines with different degrees of degradation, independently from the lane marking type (continuous, dashed, etc.) and its color (white, yellow, etc.). Furthermore, we can add plausible snow on ACDC [271] roads while preserving shadows. In addition, our model can turn Oxford Flowers dataset [225] shallow DoF photos into deep DoF ones, yielding in-focus backgrounds with sharper edges.



Figure 2.17: **Additional qualitative results on lane detection.** We train SCNN [230] and RESA [377] on images generated with our method and test on real images with damaged lane markings of IDD [313], improving performances.

PROXY TASKS

Additional proxy tasks qualitative results are shown in Fig. 2.17 for lane detection and in Fig. 2.18 for semantic segmentation. From Fig. 2.17, it is evident the robustness on degraded or even absent street lines of lane detectors trained with our translated samples. Similarly, in Fig. 2.18 we demonstrate the improved generalization of semantic segmentation models on snowy weather conditions.

2.8 DISCUSSION

In this work, we proposed a patch-based image-to-image translation model which relies on a GAN backbone trained on patches and an optional VAE to interpolate non-linearly between domains. Along with the definition of *local domains*, we introduced a dataset-based geometrical guidance strategy to ease the patches extraction process. Our few-shot method outperformed the literature on all tested metrics on several tasks (lane degradation, snow addition, deblurring), and its usability has been demonstrated on proxy tasks. In particular, our translation pipeline led to higher performances on lane detection in scenes with degraded or absent markings and on semantic segmentation in snowy conditions.

2.8. DISCUSSION

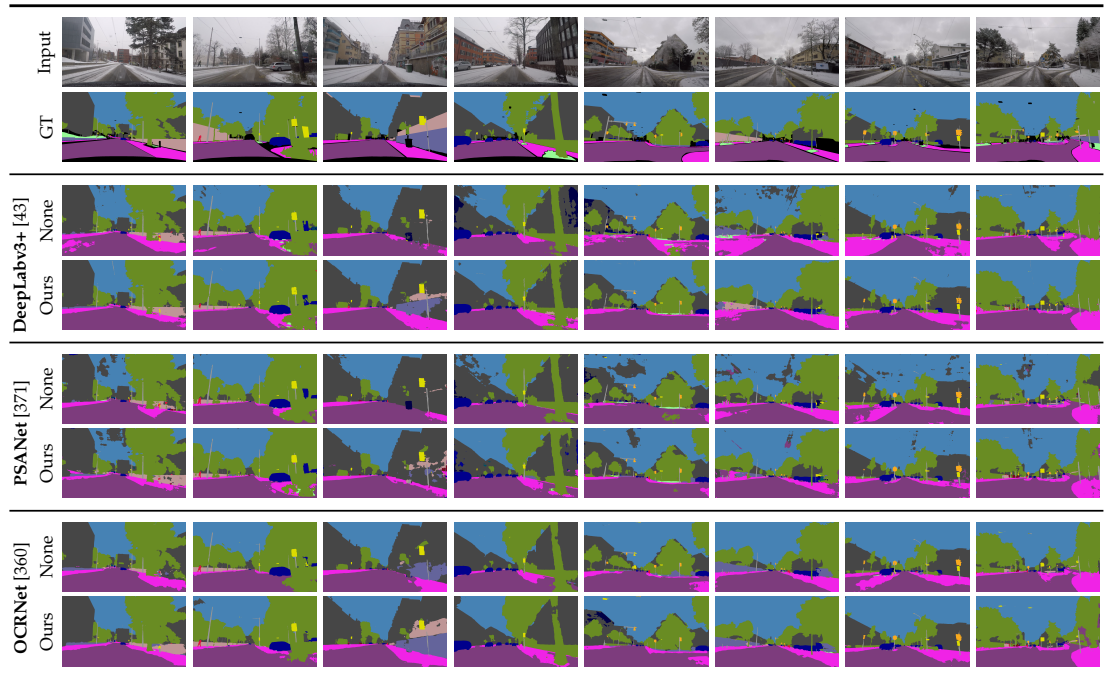


Figure 2.18: **Additional qualitative results of semantic segmentation on $\text{ACDC}_{\text{snow}}$. [271].** Using our generated images with added snow during training improves resistance of existing segmentation networks (DeepLabv3+ [43], PSANet [371] and OCRNet [360]) on snowy scenarios.

3

Point Cloud Upsampling for Enhanced 3D Data

The contributions of this chapter have been published in:

Dell'Eva, A., Orsingher, M., Bertozzi, M. (2022). Arbitrary Point Cloud Upsampling with Spherical Mixture of Gaussians. In International Conference of 3D Vision (3DV).

3.1 2D vs. 3D DATA IN AUTONOMOUS DRIVING

Self-driving cars utilize both 2D and 3D data sources, with cameras typically providing 2D images and radar or LiDAR implementing 3D sensing, as already mentioned in Sec. 1.1. Each representation has its own strengths and weaknesses.

As a primary source of 2D data, cameras are generally more cost-effective than many 3D sensing systems. They capture rich visual information, including color, texture, and other details that can be crucial in tasks like traffic sign recognition, vehicle classification, and so on. Modern cameras can detect far objects and allow to deal with fine details for better performances. Furthermore, image processing techniques and algorithms have been extensively studied, meaning the technology is mature. However, 2D data have their limitations. They cannot provide depth perception unless the camera system comes with stereoscopic setups or additional depth-sensing technologies. The quality can

3.2. 3D DATA REPRESENTATIONS

suffer a substantial degradation in various lighting conditions such as low light, glare, or direct sunlight. Moreover, 2D images can be significantly affected by occlusions or environmental factors like fog or rain.

On the other hand, 3D data excels in areas where 2D data fails. The 3D sensors provide accurate depth information, enabling precise distance measurements. They remain consistent across different lighting conditions, working efficiently both at day and night. LiDAR and radar are also less sensitive to disturbance from environmental conditions, and many systems offer a complete 360-degree view of the vehicle surroundings. Nevertheless, 3D sensing systems tend to be more expensive and they usually have a lower resolution than that of high-end cameras. They also don't capture color and texture, which can be essential for some tasks, as stated previously. Additionally, 3D data can be more challenging to process, requiring more computational resources.

In practice, a robust autonomous driving perception system integrates both 2D and 3D data. Sensor fusion aims to combine the strengths of each type while minimizing their individual weaknesses [328, 348, 123]. This multi-modal approach provides a more comprehensive view of the environment, increasing the safety and reliability of self-driving vehicles.

In our work, we propose a novel upsampling method for point clouds, a particular representation of 3D data (see next Sec. 3.2), which enables the use of arbitrary ratios, improving the quality of 3D input and ultimately enhancing the reconstruction task.

3.2 3D DATA REPRESENTATIONS

Selecting the right data representation is crucial when working with 3D deep learning systems, as each method offers various pros and cons. While there exists no canonical representation that balances computational and memory efficiency with the capability of reproducing high-resolution geometry of arbitrary topology, several representations have emerged. Those approaches have inherent trade-offs regarding efficiency, expressivity, and topological constraints. The most common are presented in this section:

- **Point clouds.** A point cloud is a collection of data points in 3D space, each defined by X, Y, and Z coordinates, as depicted in Fig. 3.1a. They can optionally include additional attributes such as RGB color. Unlike structured

data like images where pixels have clear neighboring relationships, point clouds lack regularity. This makes traditional convolution operations in deep learning infeasible. As a result, specialized models like PointNet [250, 251], have been developed to effectively process them. The main benefits are the fast rendering, which allows for quick visualization, the exact representation of 3D objects, and the possibility to execute fast modifications with simple transformations. However, the high number of points in a point cloud can lead to high memory consumption, also limiting some operations such as merging and manipulation from a computational point of view. For many applications, it is common to convert point clouds into a more structured form, like a mesh, using various surface reconstruction techniques. In our study, we perform arbitrary point cloud upsampling in order to enhance the reconstruction process.

- **Meshes.** Meshes consist of vertices (3D points) connected by edges forming faces (polygons), as shown in Fig. 3.1b. The latter can be triangles or squares. They provide additional geometric, topological, and surface-normal information compared to point clouds. This extra information proves beneficial in rendering algorithms (e.g. to compute lighting effects) and in training learning models. For instance, graph convolutional networks [147, 28, 315] deal with meshes as graphs and perform convolutional operations using vertex neighbor information [243, 319]. However, meshing interpolates the underlying point cloud geometry and can therefore represent the data only to a certain extent (curved objects are approximated). Finally, their creation can be complex and time-consuming.
- **Voxels.** Voxels (volume elements) are the extension of 2D pixels to 3D. The 3D volume is divided into a grid of smaller units, each with an occupancy value indicating if it is within the shape. Additional information like color or transparency can be associated with each element. While this representation is useful for rendering and smooth visualization, it can result in a rough approximation of the original geometry and might introduce aliasing distortions. Moreover, other downsides are the high memory requirements and the inefficiency when dealing with high-resolutions. Nevertheless, the structured grid layout of voxel models is beneficial for processing through 3D convolutional neural networks [333, 263]. An illustration of voxels can be found in Fig. 3.1c.

3.2. 3D DATA REPRESENTATIONS

- **Multi-view.** A multi-view representation of a 3D shape is a set of 2D images, each representing the 3D shape from different viewpoints. This data format is particularly convenient because it can be directly processed using 2D CNNs [293, 231]. However, multi-view representations are influenced by lighting, and capturing a comprehensive view of a 3D model might require a large number of images.
- **Special case: RGB-D.** RGB-D representations have become popular thanks to the widespread use of depth sensors. RGB-D data provides 2.5D information of the captured 3D object by combining a depth map with 2D color information. Similar to multi-view representations, this format can be readily processed with 2D CNNs [78, 387].
- **Special case: projections.** There are various projection techniques to translate the 3D data into a 2D grid with specific information. Commonly, projections into spherical and cylindrical domains have been employed for this kind of 3D data representation. These projections also simplify the processing with well-established 2D CNNs due to the resulting uniform grid structure [290, 33, 330]. Unfortunately, for complicated tasks such as dense correspondence, this format may not be ideal due to the loss of information in the projection.
- **Implicit neural representations.** Implicit representations, also known as coordinate-based representations, provide a novel way to parameterize various signals. Unlike traditional signal representations, which are typically discrete, this method utilizes a continuous function to map the domain of the signal, such as the 3D coordinates, to its corresponding output, such as the occupancy probability (Fig. 3.1d). However, these functions are usually not analytically tractable. Implicit neural representations approximate the function via a neural network. The main benefit is that the memory required to parameterize the signal is independent of spatial resolution and only scales with the complexity of the underlying signal, i.e. they have "infinite resolution". Different implicit representations help to solve distinct tasks: DeepSDF, Occupancy Networks, and similar methods [233, 36, 211, 238, 359] address the shape representation problem; SRN, NeRF, and related works [291, 212, 221, 144] deal with scene representation, in particular the novel view synthesis task; finally,

UNISURE, VolSDF, NeUS, and analogous research [228, 346, 320] aim to solve the reconstruction problem.

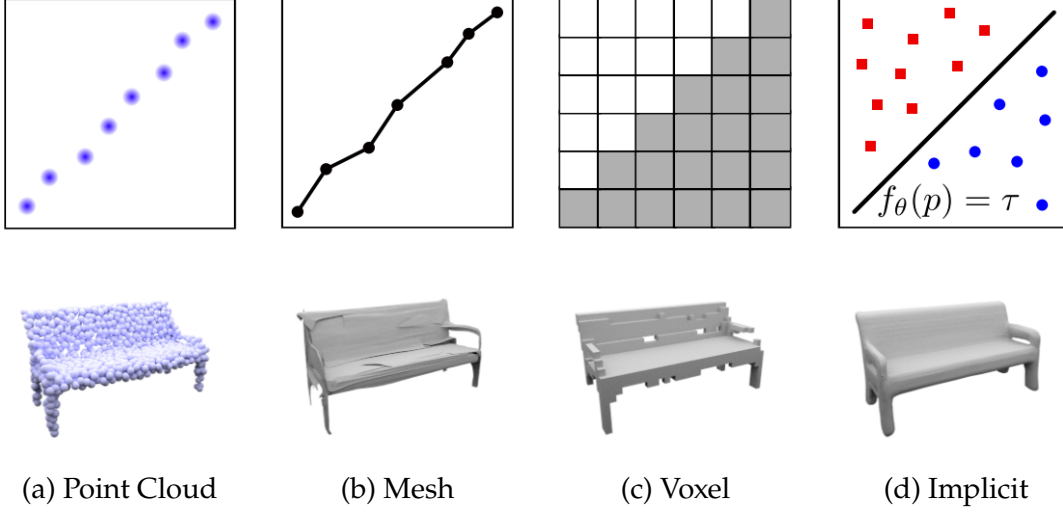


Figure 3.1: 3D representations. Source: [211].

3.3 STUDY OBJECTIVE AND CONTRIBUTIONS

As discussed before, point clouds are a common way to represent 3D data as an unordered list of points, which can be thought of as discrete samples from the underlying surface of the object. In recent years, the wide availability of low-cost scanning sensors has driven the research attention towards 3D point clouds analysis for several applications such as augmented reality, robotics and autonomous driving [375, 35, 201, 338]. However, the sparsity and the noise level in raw data from such sensors pose key challenges in point cloud processing for downstream tasks, e.g. classification, segmentation and surface reconstruction. To this end, we focus on point cloud upsampling, which consists in generating a dense and uniform set of points from a sparse and noisy input.

Building on pioneering works for neural point processing [250, 251], learning-based methods achieve state-of-the-art results in point cloud upsampling [357, 176, 252, 175] and significantly outperform classical optimization-based techniques [121, 186, 122]. One of the main limitations of current approaches is the tight coupling between the upsampling ratio and the network architecture. Typically, this value must be specified in advance and different models must be

3.3. STUDY OBJECTIVE AND CONTRIBUTIONS

re-trained from scratch for different rates. Despite recent efforts to design flexible networks with a user-defined upsampling factor at test time, existing works still limit its value to be integer and lower than a given bound [253, 202, 347], or reconstruct the whole surface with ground truth normals as an intermediate step [86].

Our main goal is to remove these limitations and to enable arbitrary upsampling from raw point clouds with a single trained model, as shown in Fig. 3.2. The key intuition of the presented method is to split the upsampling procedure into two steps: (i) firstly, the input point cloud is mapped to a probability distribution on a canonical domain; (ii) then, an arbitrary number of points can be sampled from such distribution and mapped back to the target surface.

Inspired by recent works on point cloud autoencoders [103, 177, 50], we propose to use the unit sphere as intermediate representation and we define a Spherical Mixture of Gaussians (SMOG) distribution on such domain. In this way, each input point is associated with a mixture weight and a bivariate Gaussian in spherical coordinates, whose parameters are estimated by a neural network.

The inverse mapping from the samples on the unit sphere to the desired shape is implemented by querying the decoder of a Transformer model [314] with an arbitrary number of points, which effectively decouples the network architecture and the value of the upsampling ratio. Moreover, we design both a feature extraction backbone and a residual refinement block based on local self-attention, which has proved to learn powerful representations on point cloud data [215, 358, 370]. Therefore, our main contributions can be summarized as follows:

- We present a novel approach for point cloud upsampling with arbitrary scaling factors, including non-integer values, with a single trained model.
- We propose to learn a mapping from the low-resolution input point cloud to a probability distribution on the unit sphere. This distribution can then be sampled arbitrarily and the inverse mapping is learned to generate the high-resolution output.
- We design a fully attention-based network architecture with state-of-the-art performances on multiple benchmarks and strong generalization capabilities.

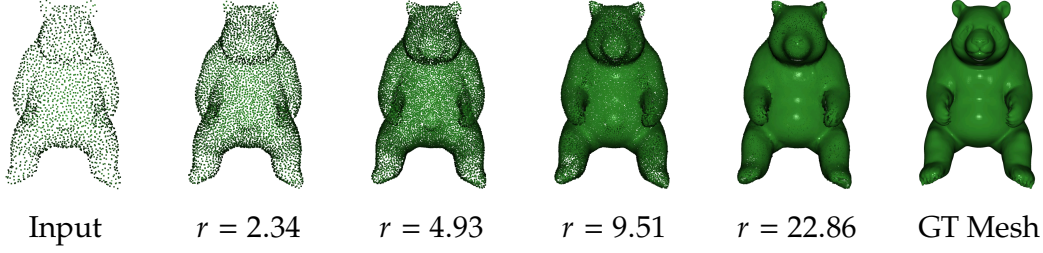


Figure 3.2: The proposed approach can perform point cloud upsampling from a sparse input with N points to a high-resolution output with $r \times N$ points. The upsampling ratio $r \in \mathbb{R}$ can be specified arbitrarily at test time, even if the model is trained a single time with $r = 4$.

3.4 RELATED WORK

Canonical Primitives. In the context of point cloud auto-encoders, a common procedure to generate the output is to feed the decoder with a global latent vector encoding the input and a canonical primitive (e.g. a 2D grid [342]), which is deformed to match the target surface. Following the insights in [103], several works use the unit sphere with *uniform* sampling as an intermediate representation [177, 50]. Recently, TearingNet [232] proposed to learn topology-friendly representations by additionally estimating pointwise offsets on the 2D domain. We take a step further by directly learning a mean vector on the unit sphere and 2D variances in spherical coordinates for each point in the input shape. This allows to define a distribution from which an arbitrary number of points are sampled and mapped back to the surface.

Learning-based Upsampling. Point cloud upsampling is an inherently ill-posed problem, since a finite number of samples correspond to many underlying surfaces and viceversa. For this reason, PU-Net [357] pioneered the idea of learning geometric priors from data and outperformed previous classical methods [122, 186, 121]. Building on this seminal idea of learning and expanding multi-scale point features, MPU [350] proposes a patch-based progressive strategy for upsampling at different levels of detail, while PU-GAN [176] casts the upsampling procedure in a generative adversarial framework. PU-GCN [252] introduces several modules built upon graph convolutional network that can be integrated into other architectures, whereas Dis-PU [175] disentangles the upsampling task into two cascaded subnetworks for dense point generation and

3.4. RELATED WORK

spatial refinement. Despite showing promising results, all these works require a fixed upsampling ratio r and train different models for varying values of r . This strategy does not adapt to real-world point clouds with different quality and it increases the training time significantly.

Arbitrary Upsampling. Recently, a few works [202, 253, 347] emerged with the goal of decoupling the upsampling ratio and the network architecture, thus achieving flexible upsampling. Meta-PU [347] employs meta-learning to predict the weights of residual graph convolution blocks dynamically for different values of r . However, the model first generates a set of $r_{max} \times N$ points, which are then downsampled to the desired ratio using farthest point sampling (FPS). MAFU [253] and PU-EVA [202] exploit the local geometry of the tangent plane at each point to sample a variable number of candidate points in its neighborhood, but they are limited to *integer* upsampling factors within a predefined range $[r_{min}, r_{max}]$. In addition, the former requires normal vectors information to be trained. On the other hand, our approach is designed to support any value of $r \in \mathbb{R}$, without restrictions. Neural Points [86] is a concurrent work that performs upsampling with unconstrained ratios by first encoding the continuous underlying surface with neural fields and then sampling an arbitrary number of points from it. Their main limitation is the requirement of ground truth normals for training, which are difficult to estimate with high accuracy, especially for noisy and sparse inputs. Conversely, our model operates in a discrete-to-discrete way on raw point clouds without normals.

Transformers for Point Clouds. The Transformer model has revolutionized both natural language processing [314] and computer vision [75], thanks to the attention mechanism at its core. Since the Transformer architecture is permutation invariant by design and thus naturally suited for 3D data, early works in the field focused on adapting its modules to point cloud processing [370, 80, 107, 208]. Attention-based methods have achieved state-of-the-art results in many 3D tasks, such as point cloud completion [358], object detection [215] and classification [339]. In this work, we propose to combine the Transformer architecture with an attention-based refinement module for point cloud upsampling. To the best of our knowledge, there is only one concurrent work on this aspect [254]. However, they solely leverage the Transformer encoder, while we also exploit the possibility of querying the Transformer decoder for arbitrary upsampling.

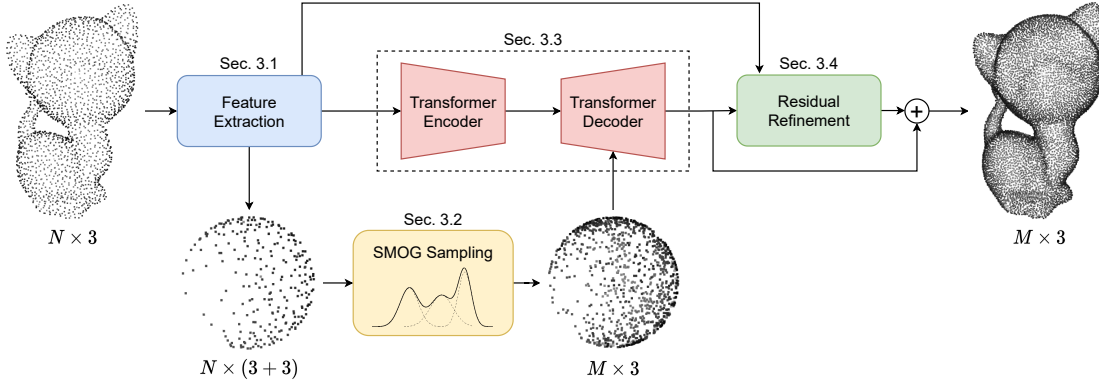


Figure 3.3: A high-level overview of the method. The low-resolution input point cloud with size $N \times 3$ is firstly mapped to a probability distribution on the unit sphere, with a mean vector and covariance matrix associated to each point. Then, this distribution can be sampled to produce the high-resolution output with size $M \times 3$, being $M = r \times N$ and r the arbitrary upsampling ratio.

3.5 METHOD

Denoting by $\mathcal{P} = \{\mathbf{p}_i \in \mathbb{R}^3\}_{i=1}^N$ the unordered sparse input point cloud of N 3D points, our objective is to generate an arbitrarily denser point set $\mathcal{Q}_r = \{\mathbf{q}_i \in \mathbb{R}^3\}_{i=1}^M$ with $M = r \times N$ points, where $r \in \mathbb{R}$ is the upsampling factor. Note that point cloud upsampling is an ill-posed task, since there is not a single feasible correct output. This means that $\mathcal{Q}$ should represent the same underlying surface $\mathcal{S}$, while not being necessarily a superset of $\mathcal{P}$. To this end, we design a fully attention-based end-to-end network for arbitrary point cloud upsampling, taking advantage of Gaussian mixture sampling and Transformer queries to enable flexible ratios. An overview of our framework is shown in Fig. 3.3.

The details are explained in the following: in Sec. 3.5.1 we present our lightweight feature extractor, while Sec. 3.5.2 describes the input mapping to a distribution on the unit sphere and the relative sampling. In Sec. 3.5.4 we show how to leverage the Transformer model to obtain a flexible number of output points. Lastly, in Sec. 3.5.5 we illustrate the refinement of the 3-dimensional coordinates to generate the final upsampled point cloud.

3.5. METHOD

3.5.1 FEATURE EXTRACTION

The first step of our pipeline is to extract point-wise features from the $N \times 3$ input point cloud to obtain a $N \times D$ feature map. Differently from previous methods [252, 357, 176] that employ either PointNet [250, 251] or DGCNN [326] as backbone, we design a lightweight network based on Point Transformer [370]. For each input point $\mathbf{p}_i \in \mathcal{P}$, its associated feature is computed by the Point Transformer Layer (PTL) as follows:

$$\mathbf{f}_i = \sum_{\mathbf{p}_j \in \mathcal{N}(\mathbf{p}_i)} \rho(\gamma(\beta(\mathbf{p}_i) - \psi(\mathbf{p}_j) + \delta)) \odot (\alpha(\mathbf{p}_j) + \delta) \quad (3.1)$$

where $\mathcal{N}(\mathbf{p}_i)$ is the set of k nearest neighbors of $\mathbf{p}_i$, $\delta = \eta(\mathbf{p}_i - \mathbf{p}_j)$ is the positional encoding and the symbol $\odot$ denotes the element-wise product. The mappings α , β and ψ are simple linear layers, γ and η are two-layers MLP, while ρ is the softmax function. Note that in the context of the classical interpretation of the Transformer model [314], α generates the *values*, β the *queries* and ψ the *keys*.

3.5.2 SPHERICAL MIXTURE OF GAUSSIANS SAMPLING

The core intuition of our approach is to map the input point cloud into a probability distribution on a canonical domain which can be conveniently sampled. To this end, we estimate the parameters of a Gaussian Mixture Model (GMM) on the unit sphere from the deep features extracted by the PTL. The choice of this domain over a 2D square is motivated by the fact that the unit sphere is a manifold without boundaries, which avoids the truncation of the Gaussian distributions in the GMM.

More specifically, we define a K components GMM $\Gamma = \{w_i, \mu_i, \Sigma_i\}_{i=1}^K$, where w_i , μ_i and Σ_i are the mixture weight, mean and covariance of the i -th Gaussian and $K = N$. Therefore, the likelihood of a point $\mathbf{z}$ on the unit sphere is given by a weighted combination of individual components $u_i(\mathbf{z})$:

$$u_\Gamma(\mathbf{z}) = \sum_{i=1}^N w_i u_i(\mathbf{z}) \quad (3.2)$$

Each component is weighted equally (i.e. $w_i = \frac{1}{N}$) and its parameters (μ_i, Σ_i) are estimated by a MLP ξ :

$$(\mu_i, \Sigma_i) = \xi(\mathbf{f}_i) \quad (3.3)$$

The MLP predicts the mean vector μ_i in Cartesian coordinates and the covariance matrix Σ_i in spherical coordinates. In particular, the covariance matrix is built by exploiting the constraint of symmetry:

$$\Sigma_i = \begin{bmatrix} \sigma_\theta^2 & \sigma_{\theta\phi} \\ \sigma_{\theta\phi} & \sigma_\phi^2 \end{bmatrix} \quad (3.4)$$

where σ_θ^2 , σ_ϕ^2 and $\sigma_{\theta\phi}$ are the outputs of the covariance head of the MLP corresponding to the azimuth angle $\theta \in [0, 2\pi]$ and the elevation angle $\phi \in [0, \pi]$, respectively. In order to generate a dense output point cloud, we sample an arbitrary number of points from the distribution and learn the inverse mapping from the spherical domain to the target surface. This process is illustrated in Fig. 3.4.

3.5.3 DETAILS ON SMOG SAMPLING

In this section, we discuss the implementation details of the sampling process from the SMOG distribution. More specifically, we have to ensure that the covariance matrix Σ is positive semi-definite, in order for it to be well-defined. The simplest way to impose this condition is to clamp the covariance value $\sigma_{\theta\phi}$ between the following bounds:

$$-\sigma_\theta\sigma_\phi \leq \sigma_{\theta\phi} \leq \sigma_\theta\sigma_\phi \quad (3.5)$$

To prove this, consider a real symmetric 2×2 matrix:

$$A = \begin{bmatrix} a & b \\ b & c \end{bmatrix} \quad (3.6)$$

Such a matrix is positive semi-definite if and only if its eigenvalues are non-negative. The eigenvalues of a 2×2 matrix can be computed in closed form as the solutions to the following equation:

$$\det(A - \lambda I) = (a - \lambda)(c - \lambda) - b^2 = 0 \quad (3.7)$$

3.5. METHOD

We can immediately apply the well-known formula for solving quadratic equations:

$$\lambda_{1,2} = \frac{a + c \pm \sqrt{(a + c)^2 - 4(ac - b^2)}}{2} \quad (3.8)$$

The argument of the square root is always non-negative, as it can be rewritten as the sum of two positive numbers:

$$4b^2 + (a - c)^2 \geq 0 \quad (3.9)$$

Since $a > 0$ and $c > 0$ by construction, the first solution is also guaranteed to be non-negative as the sum of three non-negative components:

$$\lambda_1 = \frac{a + c + \sqrt{(a + c)^2 - 4(ac - b^2)}}{2} \geq 0 \quad (3.10)$$

Let's analyze the second solution:

$$\lambda_2 = \frac{a + c - \sqrt{(a + c)^2 - 4(ac - b^2)}}{2} \geq 0 \quad (3.11)$$

It can be rewritten as follows:

$$a + c \geq \sqrt{(a + c)^2 - 4(ac - b^2)} \quad (3.12)$$

Since both sides of the inequality are non-negative, we can square them and we can expand the right-hand side:

$$a^2 + 2ac + c^2 \geq a^2 + c^2 - 2ac + 4b^2 \quad (3.13)$$

With simple computations, this becomes:

$$4ac \geq 4b^2 \implies |b| \leq \sqrt{ac} \implies -\sqrt{ac} \leq b \leq \sqrt{ac} \quad (3.14)$$

Note that this finding is confirmed by noting that the correlation value $\rho_{\theta\phi}$ is bounded between $[-1, 1]$ and by definition:

$$\rho_{\theta\phi} = \frac{\sigma_{\theta\phi}}{\sqrt{\sigma_{\theta}^2 \sigma_{\phi}^2}} \implies -1 \leq \frac{\sigma_{\theta\phi}}{\sqrt{\sigma_{\theta}^2 \sigma_{\phi}^2}} \leq 1 \implies -\sigma_{\theta}\sigma_{\phi} \leq \sigma_{\theta\phi} \leq \sigma_{\theta}\sigma_{\phi} \quad (3.15)$$

From an implementation point of view, we ensure this condition with the `torch.clamp` function. Even if this naive approach sets the gradients and the related update signal to zero outside the acceptance region, we have found experimentally that very few values lie outside this range during training. An alternative approach for imposing the same constraint is to decompose the covariance matrix as follows:

$$\Sigma = \mathbf{R}(\alpha)\mathbf{D}\mathbf{R}(\alpha)^\top \quad (3.16)$$

where $\mathbf{R}(\alpha)$ is the 2D rotation matrix for a given angle α and $\mathbf{D}$ is a diagonal matrix with variances as diagonal elements. The resulting covariance matrix is symmetric and positive definite as required, with the additional advantage of predicting a continuous value for the angle α as the output of the network, instead of clamping and zeroing the gradients. We have implemented and tested this strategy, but we did not notice any significant improvement with respect to the previous approach.

3.5.4 TRANSFORMER MODEL

The second core intuition which underpins our work is the possibility of querying the Transformer model [314] with an arbitrary number of points. Inspired by the 3DETR architecture [215], the set of $N \times D$ features produced by our backbone is fed to an encoder to produce a new feature map of dimension $N \times D$. The Transformer encoder has a single layer with a four-headed attention and an MLP with two layers. These features, along with a set of $r \times N$ queries, are processed by the two-layers Transformer decoder to generate $r \times N$ 3-dimensional Euclidean coordinates, which represent the coarse upsampled point cloud. The queries are obtained by embedding with Fourier positional encoding [300] the points sampled from the unit sphere manifold.

3.5.5 RESIDUAL REFINEMENT

The coarse output from the Transformer decoder might be noisy, non-uniform distributed and have several outliers. Motivated by other works [175, 253], we design an attention-based residual refinement with a similar architecture as the feature extractor described in Sec. 3.5.1. In particular, we employ a single Point Transformer Block (PTB) [370] composed by a PTL, two linear

3.5. METHOD

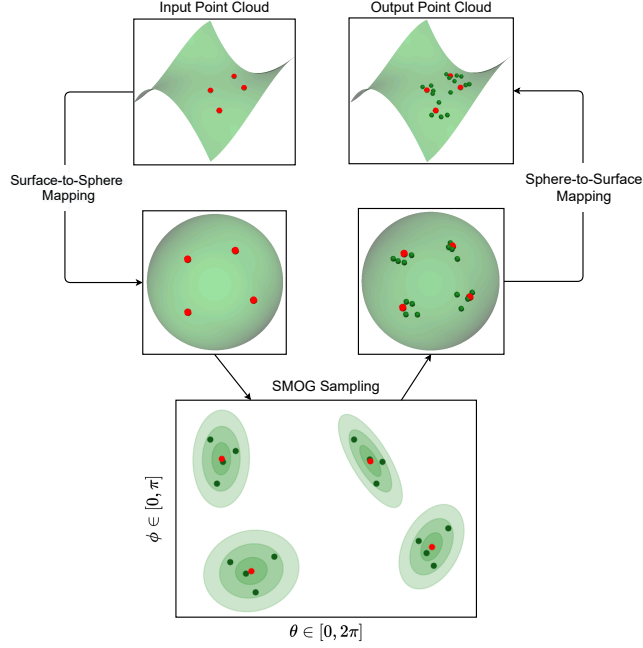


Figure 3.4: The sampling process shown in detail. Red points are the Gaussian means on the unit sphere associated to each input point, while green points are the arbitrary samples from the SMOG. Red points on the right branch are maintained for visualization.

layers, a ReLU activation function and a residual connection. Differently from Eq. 3.1, in this case the PTL takes as input the local features extracted from the backbone, while the positional encodings are computed with the 3D coarse coordinates. The D -dimensional vector produced by the PTB is finally projected to a 3-dimensional space with a two-layers MLP that computes the residual for each point, which is added to the coarse prediction to obtain the refined result.

3.5.6 LOSS FUNCTION

Let $\tilde{\mathcal{Q}}_r = \{\tilde{\mathbf{q}}_i \in \mathbb{R}^3\}_{i=1}^{rN}$ be the coarse prediction generated by the Transformer with upsampling rate r , $\mathcal{Q}_r = \{\mathbf{q}_i \in \mathbb{R}^3\}_{i=1}^{rN}$ the refined prediction and $\mathcal{Y}_r = \{\mathbf{y}_i \in \mathbb{R}^3\}_{i=1}^{rN}$ the ground-truth upsampled point cloud. For training, we adopt a similar strategy as in [86] to define a distance between two generic point clouds $\mathcal{X} = \{\mathbf{x}_i \in \mathbb{R}^3\}_{i=1}^X$ and $\mathcal{Z} = \{\mathbf{z}_i \in \mathbb{R}^3\}_{i=1}^Z$ as:

$$d_{\Pi}(\mathcal{X}, \mathcal{Z}) = \frac{1}{X} \sum_{i=1}^X \|\mathbf{x}_i - \Pi(\mathbf{x}_i, \mathcal{Z})\|_2^2 \quad (3.17)$$

where $\Pi(\cdot, \cdot)$ is the projection of a 3D point to a point cloud. This term is computed as the weighted combination of the nearest points to $\mathbf{x}$ in $\mathcal{Z}$, with indices $\mathcal{N}(\mathbf{x}, \mathcal{Z})$:

$$\Pi(\mathbf{x}, \mathcal{Z}) = \frac{\sum_{k \in \mathcal{N}(\mathbf{x}, \mathcal{Z})} w_k \mathbf{z}_k}{\sum_{k \in \mathcal{N}(\mathbf{x}, \mathcal{Z})} w_k} \quad (3.18)$$

The weights w_k are given by:

$$w_k = e^{-\alpha \|\mathbf{x} - \mathbf{z}_k\|_2^2}, \quad k \in \mathcal{N}(\mathbf{x}, \mathcal{Z}) \quad (3.19)$$

with $\alpha = 10^3$. The corresponding loss is then defined by the following bidirectional sum:

$$\mathcal{L}_\Pi(\mathcal{X}, \mathcal{Z}) = d_\Pi(\mathcal{X}, \mathcal{Z}) + d_\Pi(\mathcal{Z}, \mathcal{X}) \quad (3.20)$$

Our arbitrary upsampling network is trained with the sum of three losses:

$$\mathcal{L} = \mathcal{L}_\Pi(\tilde{\mathcal{Q}}_4, \mathcal{Y}_4) + \mathcal{L}_\Pi(\mathcal{Q}_4, \mathcal{Y}_4) + \mathcal{L}_{ACD}(\tilde{\mathcal{Q}}_1, \mathcal{P}) \quad (3.21)$$

where the last term is the Augmented Chamfer Distance (ACD) between the reconstructed point cloud and the input, defined for two generic point clouds as follows:

$$\mathcal{L}_{ACD}(\mathcal{X}, \mathcal{Z}) = \max \left\{ \frac{1}{X} \sum_{\mathbf{x} \in \mathcal{X}} \min_{\mathbf{z} \in \mathcal{Z}} \|\mathbf{x} - \mathbf{z}\|_2^2, \frac{1}{Z} \sum_{\mathbf{z} \in \mathcal{Z}} \min_{\mathbf{x} \in \mathcal{X}} \|\mathbf{x} - \mathbf{z}\|_2^2 \right\} \quad (3.22)$$

In practice, each training iteration consists of two forward passes and a single backward pass. During the upsampling forward pass, the Transformer decoder is queried with $4 \times N$ points sampled from the SMOG and the projection loss components for both the coarse and refined upsampled outputs are computed. On the other hand, in the reconstruction phase, the estimated means of the SMOG components are fed to the decoder and the ACD with respect to the input point cloud is evaluated. This additional term is required to learn the proper positions of the Gaussian means on the unit sphere, which acts effectively as an intermediate probabilistic representation of the input shape. The ablation studies in Sec. 3.6.6 prove this insight with numerical evidence.

3.6 EXPERIMENTS

3.6.1 SETTINGS

Datasets. In order to compare the proposed APU-SMOG with state-of-the-art *fixed-ratio* methods, we employ the challenging PU1K dataset [252], which contains 1147 synthetic 3D models of various shapes, split into 1020 training samples and 127 testing samples. Consistently with existing literature [252, 175, 176, 350], 50 patches are extracted from each training shape with 256 points as input to the model and 1024 points as ground truth ($r = 4$). During testing, 2048 points are sampled from the original mesh with Poisson disk sampling. We employ a similar strategy as in [356] to extract overlapping patches with 256 geodesically close points. The network predicts the upsampled outputs at patch level and the final result is given by combining the overlapping patches with FPS to obtain 8192 points.

Moreover, we evaluate our approach against *flexible-ratio* methods on the widely used PU-GAN dataset [176], which is composed by 147 shapes collected from the PU-Net [357], MPU [350] and Visionair [316] repositories. The same patch-based training and testing procedure is followed with 2048 points as input, but ground truth point clouds with different sizes are generated to adapt to the values of the scaling factor r . Finally, we test the generalization capabilities of our approach on real-world LiDAR point clouds from the KITTI dataset [91].

Evaluation Metrics. Following previous works, the results are quantitatively evaluated with the Chamfer Distance (CD), Hausdorff Distance (HD) and Point-to-Surface (P2F) distance metrics. The CD is the sum of squared distances between nearest neighbor correspondences of the input and ground truth point clouds, while the HD effectively measures the influence of outliers in the predicted results. On the other hand, the P2F distance is computed against the underlying surface, thus estimating the quality of the upsampled point cloud as an approximation of the real shape. All the metrics are averaged on the whole test set and a lower value indicates better upsampling performance.

Comparison Methods. For the task of upsampling with a fixed ratio $r = 4$ on the PU1K dataset, we provide quantitative comparison against the state-of-the-art models PU-Net [357], MPU [350], PU-GAN [176], PU-GCN [252] and Dis-

PU [175]. On the other hand, the flexible methods MAFU [253], PU-EVA [202] and Neural Points [86] are used as baselines for arbitrary upsampling on the PU-GAN dataset. For a fair comparison, we used the official pre-trained models when available and re-trained the other ones with the official published code.

Implementation Details. Our framework is implemented in PyTorch [236]. The features dimension is set to $D = 128$. The MLPs in the Transformer model has hidden dimension equal to 64 in the encoder and 128 in the decoder. For the local feature extractor and the refinement module, the number of neighboring points are set to 32 and $4r$, respectively. The projection components in the loss function are weighted with $\lambda_1 = \lambda_2 = 0.01$, while the reconstruction component weight is set to $\lambda_3 = 1$. In the $\Pi(\cdot, \cdot)$ computation, k is equal to 4. The model is trained with a batch size of 64 for 100K iterations on a single V100 GPU with 32GB of RAM, using random rotation and perturbation as data augmentation techniques to avoid overfitting. The AdamW [198] optimizer is used with a learning rate decay following a cosine schedule [199] from $5e-4$ to $1e-6$, a weight decay of 0.1 and a gradient L_2 norm clipping of 0.1.

3.6.2 QUANTITATIVE RESULTS

The quantitative evaluation of the method on the PU1K test set and a fixed upsampling ratio $r = 4$ for point clouds with $N = 2048$ points is presented in Tab. 3.1. It can be noticed that we achieve the lowest HD value, indicating that APU-SMOG performs upsampling with fewer outliers with respect to state-of-the-art models, as well as the lowest P2F distance, showing a better approximation of the underlying surface. Our approach falls behind Dis-PU [175] in terms of the CD metric by a slight margin. Nevertheless, this quantity measures the consistency of the result with the ground truth *point cloud*, rather than with the *target shape*: two different sets of points sampled from the same mesh have a non-zero CD, despite belonging to the same surface.

Furthermore, the numerical performances as a function of the ratio r on the PU-GAN test set are provided in Tab. 3.2. In order to be able to compare against state-of-the-art flexible methods [253, 202], we constrain r to be an integer value smaller than 16, even if we can generate predictions with any $r \in \mathbb{R}$. Our approach achieves the best CD value, as well as the best HD value by a significant margin, along the whole spectrum of ratios. Note that both MAFU [253] and

3.6. EXPERIMENTS

Method	CD↓	HD↓	P2F μ ↓	P2F σ ↓
PU-Net [357]	1.155	11.626	4.834	6.799
MPU [350]	0.935	10.298	3.551	5.971
PU-GAN [176]	0.885	16.539	3.717	5.746
PU-GCN [252]	0.584	5.822	2.499	4.441
Dis-PU [175]	0.511	<u>4.104</u>	<u>2.013</u>	<u>2.926</u>
Ours	<u>0.528</u>	2.549	1.667	2.075

Table 3.1: Quantitative comparison with state-of-the-art methods on the PU1K dataset and $r = 4$. The units are all 10^{-3} and lower is better. Best and second results are **bold** and underlined.

Method		$r = 4$			$r = 8$			$r = 12$			$r = 16$		
		CD↓	HD↓	P2F↓	CD↓	HD↓	P2F↓	CD↓	HD↓	P2F↓	CD↓	HD↓	P2F↓
Fixed	PU-GAN [176]	0.274	4.694	1.943	0.489	6.985	2.621	0.233	6.093	2.548	0.209	6.055	2.556
	PU-GCN [252]	0.304	2.656	2.541	0.256	4.175	2.825	0.204	4.157	2.737	0.195	4.176	2.716
	Dis-PU [175]	0.360	5.133	2.868	0.352	7.028	3.338	0.291	6.694	3.258	0.271	6.645	3.240
Flexible	MAFU [253]	0.322	2.116	1.721	0.195	2.389	2.037	0.164	2.392	2.034	0.158	2.367	1.971
	NePs [86]	0.368	4.556	<u>1.875</u>	0.254	10.146	1.928	0.203	10.018	1.922	0.159	9.263	1.957
	PU-EVA [202]	0.394	7.676	2.915	0.322	7.951	3.148	0.290	8.191	3.234	0.286	8.390	3.269
	Ours	0.276	1.909	2.634	0.194	1.628	2.613	0.162	1.626	2.635	0.149	1.948	2.666

Table 3.2: Quantitative comparison with state-of-the-art methods on the PU-GAN dataset and flexible upsampling ratios. The units are all 10^{-3} and lower is better. Best and second results *among flexible methods* are **bold** and underlined. Fixed methods are included as reference.

Neural Points [86] have a lower P2F metric since they require ground truth normals at training time, which helps in finding the correct surface. On the other hand, our approach consistently outperforms PU-EVA [202], which is trained from raw point clouds. For completeness, we include the comparison against *fixed-ratio* models as reference. To generate predictions for $r \in \{8, 12, 16\}$, each network is queried iteratively with $r = 4$ and the desired number of points is obtained with FPS.

3.6.3 QUALITATIVE RESULTS

Fig. 3.5 shows qualitative upsampling results in comparison with state-of-the-art methods. These results have been obtained under the same settings as Tab. 3.1, namely with all the models trained on the PU1K dataset with fixed upsampling ratio $r = 4$ for input point clouds having size $N = 2048$. Close-

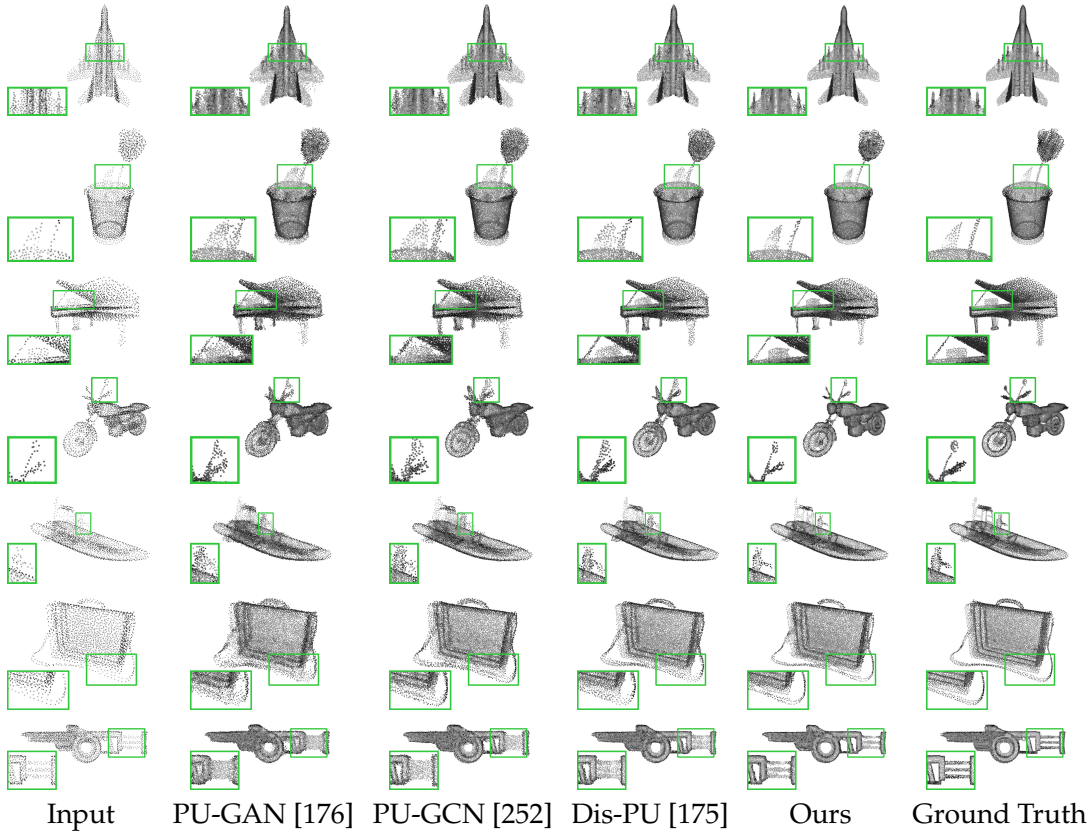


Figure 3.5: Qualitative comparison with state-of-the-art methods on the PU1K dataset. Inputs with 2048 points (left) are upsampled to 8192 points (right), with upsampling ratio $r = 4$. Details are best viewed when zoomed in.

up views show that APU-SMOG is particularly effective in preserving fine-grained structures, such as the piano’s pole and the motorcycle mirror, and disambiguating complex shape (see the bag’s handle). It can be noticed that other models fail to distinguish between different details of the surface and tend to merge them together, thus producing noisy point clouds. Moreover, the proposed attention-based residual refinement block generates refined outputs with fewer outliers (e.g. the plane motors). Finally, the visualization of Fig. 3.6, obtained with $N = 256$, shows that the model is learning a meaningful mapping, preserving in that case the cup hole.

3.6.4 GENERALIZATION AND ROBUSTNESS

We conduct further experiments to demonstrate the robustness and generalization capabilities of APU-SMOG. Firstly, we provide qualitative results on

3.6. EXPERIMENTS

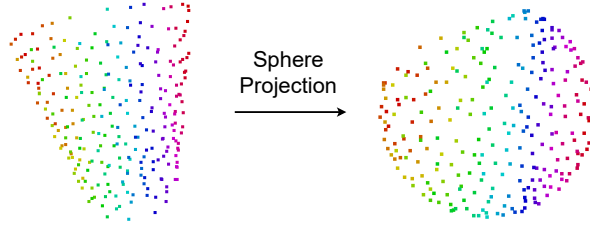


Figure 3.6: Visualization of the learned spherical representation. The model learns a meaningful mapping, preserving the cup hole.

real-world point clouds from the KITTI dataset [91], without any fine-tuning. This task is particularly challenging, since street-level LiDAR data with noise and occlusions are very different from synthetic training samples. Fig. 3.7 shows the generalization power of our approach on different urban elements such as cars, trucks and pedestrians.

Since the previous qualitative results in Fig. 3.5 have been generated with point clouds having $N = 2048$ points, we show in Fig. 3.8 that the predictions of our method closely follow the underlying surfaces for a wide variety of input sizes. Even for the sparsest input with 256 points, thin structures such as the horse’s legs are upsampled correctly. In order to simulate real noisy point clouds from scanning sensors, Fig. 3.9 shows the upsampling results for three different levels of additive Gaussian noise. It can be noticed that the duck shape is successfully maintained for both the clean and the corrupted inputs.

Finally, the key novelty of our approach is the possibility to specify an arbitrary upsampling factor at test time, thus producing consistent results with the desired output resolution. This flexibility is shown in Fig. 3.2.

3.6.5 SURFACE RECONSTRUCTION

In this section, we present the qualitative results of our approach when the upsampled point clouds are used to extract a continuous mesh. This experiment is performed with the model trained on the PU1K dataset [252] with input size $N = 2048$, $r = 4$ and the open-source implementation of Poisson Surface Reconstruction (PSR) available in the Open3D library [378]. Fig. 3.10 shows that the outputs of our method allow to reconstruct better shapes with respect to other state-of-the-art methods (e.g. the index finger in the hand and the human arm in the statue).

Moreover, we provide additional surface reconstruction results from APU-

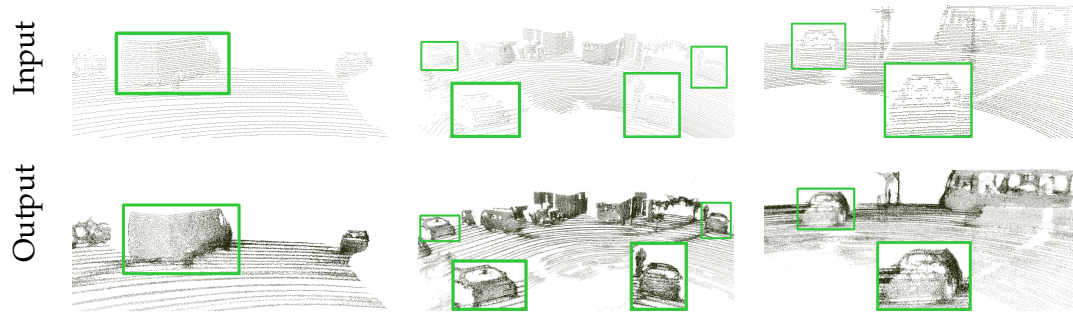


Figure 3.7: Generalization to real-world point clouds from the KITTI dataset [91]. APU-SMOG can upsample successfully various urban instances, including fine-grained structures such as the car window. Details are best viewed when zoomed in.

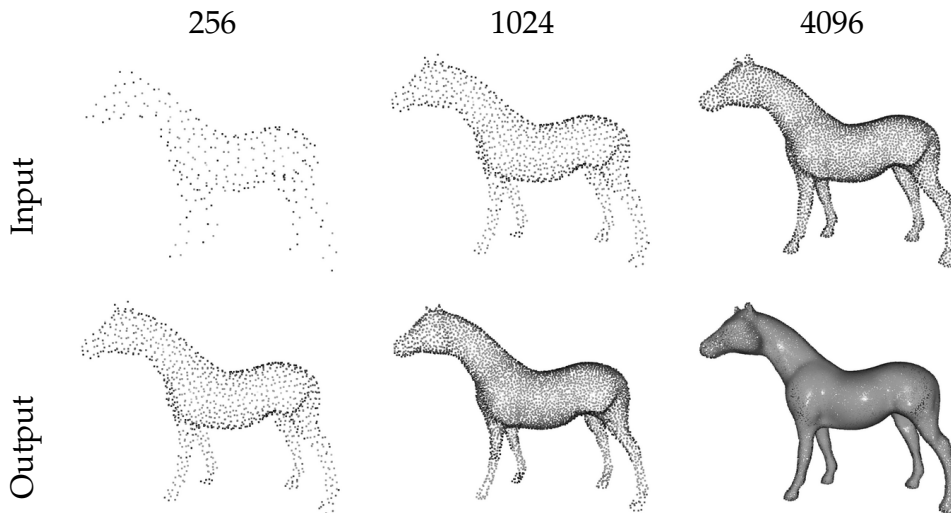


Figure 3.8: Effect of input point cloud size on the upsampling results.

3.6. EXPERIMENTS

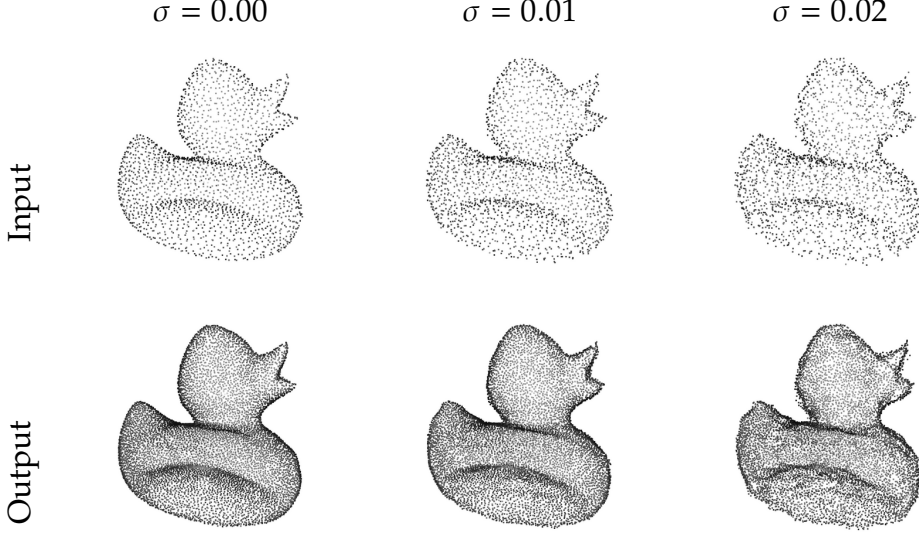


Figure 3.9: Effect of additive noise on the upsampling results.

SMOG as a function of the upsampling ratio in Fig. 3.11. In this case, the input point cloud has size $N = 1024$ and the ratios are chosen randomly in a wide range to further demonstrate the ability of handling arbitrary factors $r \in \mathbb{R}$. Note that the ground truth mesh is the actual synthetic shape from the PU1K dataset, while the ground truth point cloud is generated for reference by Poisson disk sampling. The quality of the resulting mesh consistently improves with increasing values of the upsampling ratio, thus proving the effectiveness of our method for the downstream 3D reconstruction task.

3.6.6 ABLATION STUDIES

We perform a set of ablation studies to evaluate the contribution of each module to the proposed pipeline. The presented results in Tab. 3.3 have been obtained with different versions of our model trained on the PU1K dataset with fixed upsampling ratio $r = 4$.

In order to measure the influence of the SMOG representation, we replace it with a FoldingNet-like [342] strategy, i.e. sampling the unit sphere uniformly. The quantitative results confirm that our approach is able to generate better predictions, as it adaptively learns a local probability distribution around each point.

Furthermore, we investigate the influence of the number of GMM compo-

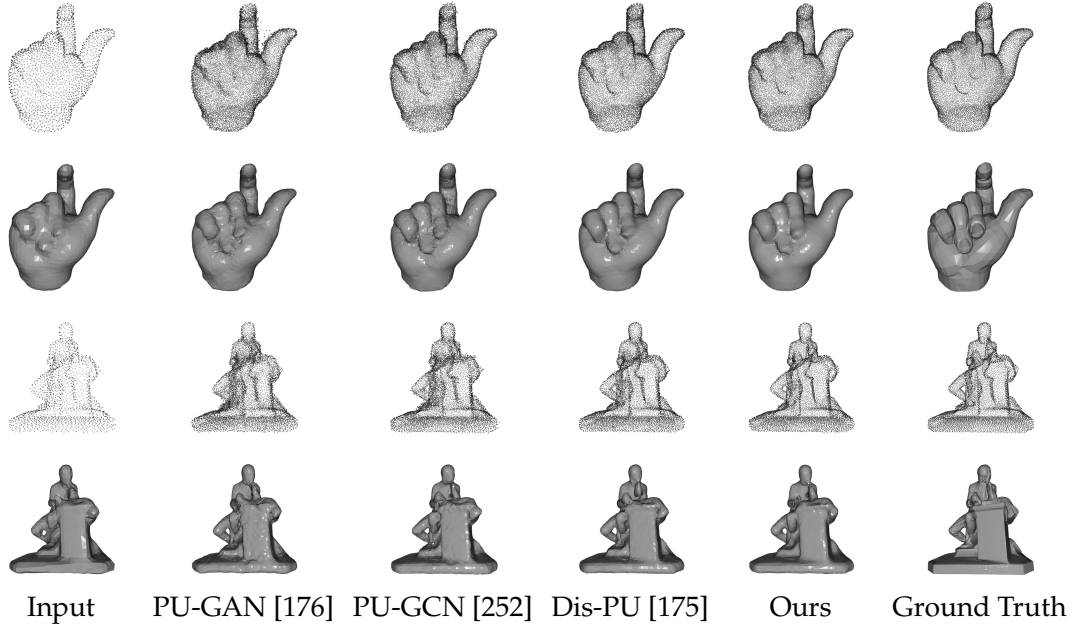


Figure 3.10: Qualitative comparison with state-of-the-art methods when the upsampled point clouds are used to compute a mesh with PSR.

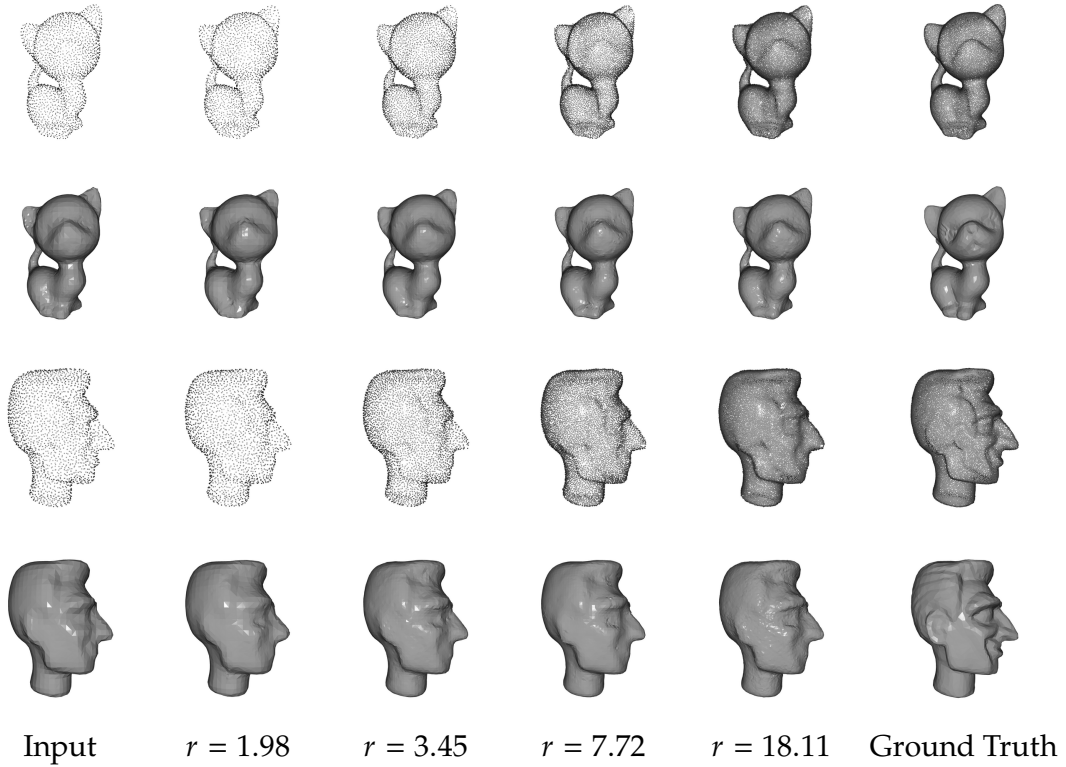


Figure 3.11: Qualitative comparison with arbitrary upsampling ratios when the point clouds are used to compute a mesh with PSR.

3.6. EXPERIMENTS

Ablation	CD↓	HD↓	P2F↓
FoldingNet-like [342]	0.558	2.761	1.700
$N/4$ SMOG components	0.564	2.803	1.686
w/o refinement	0.572	8.536	2.182
w/o rec. loss, $\mathcal{L}_{\Pi}$ ups.	0.561	2.843	1.726
w/o rec. loss, $\mathcal{L}_{ACD}$ ups.	0.816	6.459	2.694
Ours	0.528	2.549	1.667

Table 3.3: Ablation studies on different versions of our model. The units are all 10^{-3} and **bold** denotes the best performance.

nents on the output point clouds, reducing it to $K = N/4$. The numerical evaluation exhibits decent performances, suggesting that the model associates each Gaussian distribution to a local neighborhood of points. However, having a single component for each input point leads to the best results overall.

The third row in Tab. 3.3 proves the effectiveness of the attention-based residual refinement step. The significantly higher HD value indicates that the raw output from the Transformer decoder contains several outliers, which are correctly positioned closer to the surface by this module.

Moreover, we train our model without the reconstruction loss and notice a performance drop. This implies that $\mathcal{L}_{ACD}$ is a strong bias towards learning the correct positions of the Gaussian means on the unit sphere. Finally, the advantage of the projection loss for upsampling over ACD is shown in the last row. The remarkable difference in all the metrics justifies the choice of $\mathcal{L}_{\Pi}$ in the final design.

3.6.7 ROBUSTNESS TO INPUT SIZE

We also study the effect of input point cloud size on the upsampling results from a different point of view. Whereas we previously show the differences in the upsampled point clouds for a *fixed* ratio $r = 4$, in Fig. 3.12 the output size is maintained fixed at $M = 8192$ and the upsampling factor varies in $r \in \{16, 8, 4\}$ for different input sizes. It can be seen that APU-SMOG achieves satisfying results also for the sparsest input $N = 512$. This robustness, combined with the flexibility of our method, provides a full control on the input and output sizes for real-world upsampling scenarios, where the number of points might need to

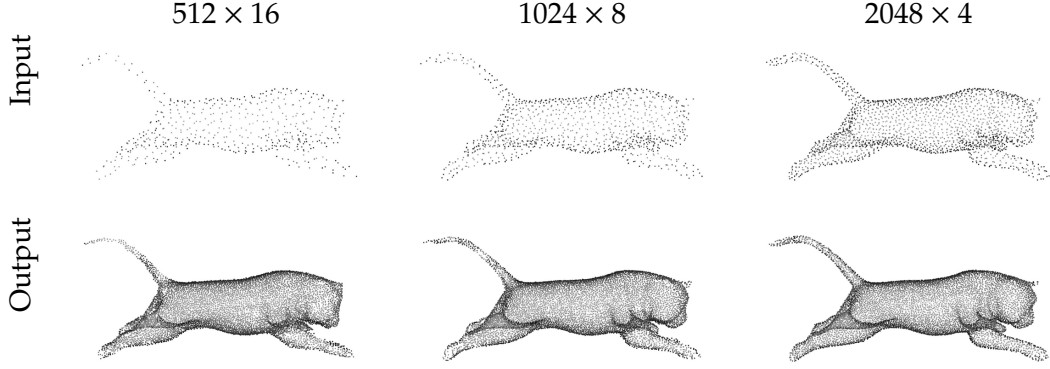


Figure 3.12: Effect of input point cloud size on the upsampling results while keeping the output size fixed.

adapt to computational and bandwidth constraints.

3.6.8 ADDITIONAL QUALITATIVE RESULTS

In this section, we provide additional qualitative results of our method under different settings. Firstly, several upsampled point clouds from the PU1K dataset with $N = 2048$ and $r = 4$ are shown in Fig. 3.13. These further examples confirm that APU-SMOG outperforms state-of-the-art methods in preserving fine-grained structures and reducing outliers.

Moreover, we investigate the upsampling results on the PU-GAN dataset [176] against concurrent flexible architectures for different ratios $r \in \{4, 8, 12, 16\}$. Close-up views in Fig. 3.14 show that our approach produces cleaner shapes along the whole range of upsampling factors, even when compared with methods requiring ground truth normals at training time [253, 86].

Finally, Fig. 3.15 shows additional examples on real-world objects point clouds from the ScanObjectNN dataset [312]. This figure is complementary to the visualization of upsampling results on real LiDAR data from KITTI [91]. In both cases, our model is robust to domain shifts, without any fine-tuning.

3.7 DISCUSSION

In this study, we present a novel approach for point cloud upsampling with arbitrary scaling factors. A Transformer-based architecture is designed to decouple the upsampling process in two key steps: (i) firstly, the sparse input is

3.7. DISCUSSION

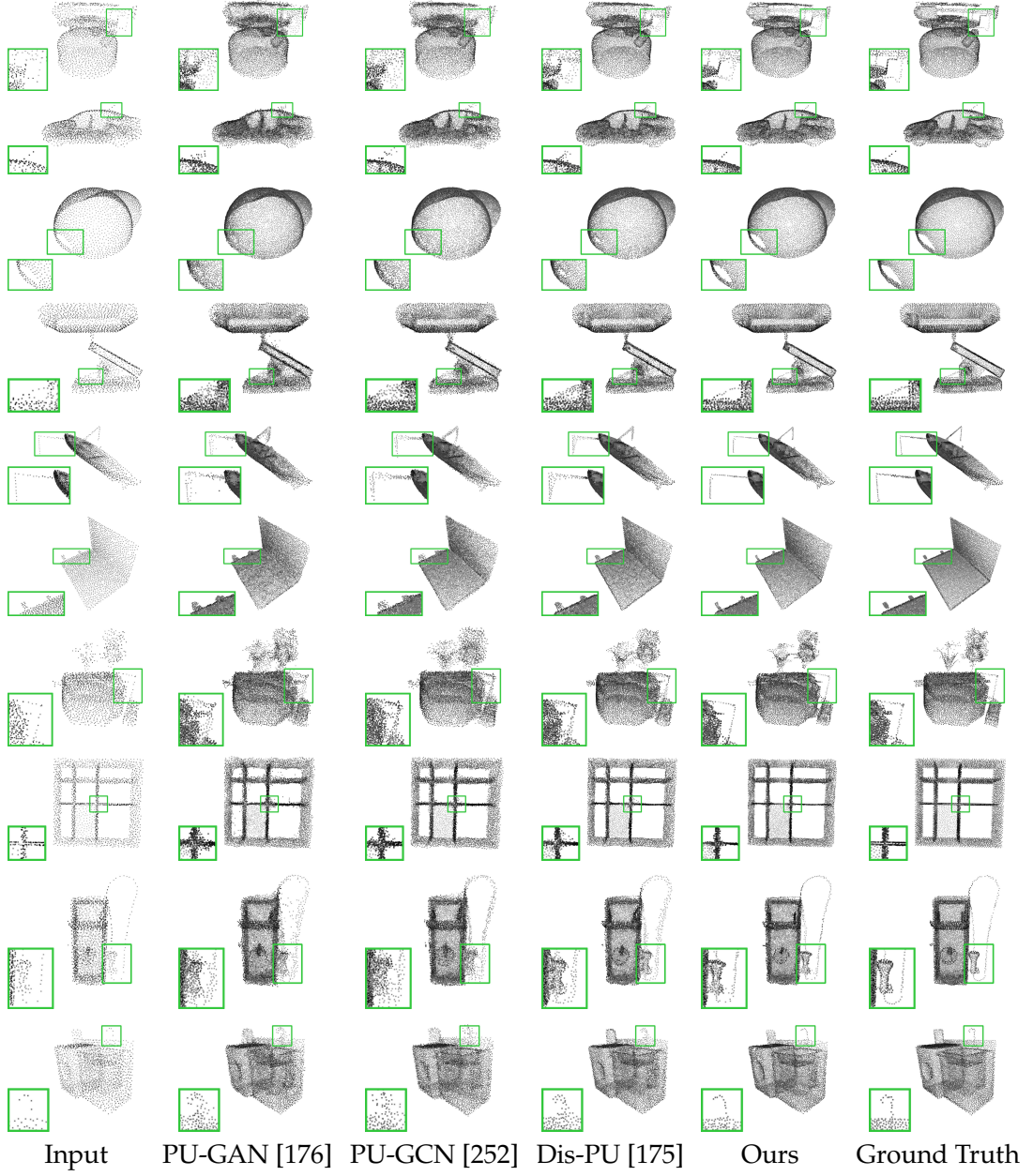


Figure 3.13: Qualitative comparison with state-of-the-art methods on the PU1K dataset. Inputs with 2048 points (left) are upsampled to 8192 points (right), with upsampling ratio $r = 4$. Details are best viewed when zoomed in.

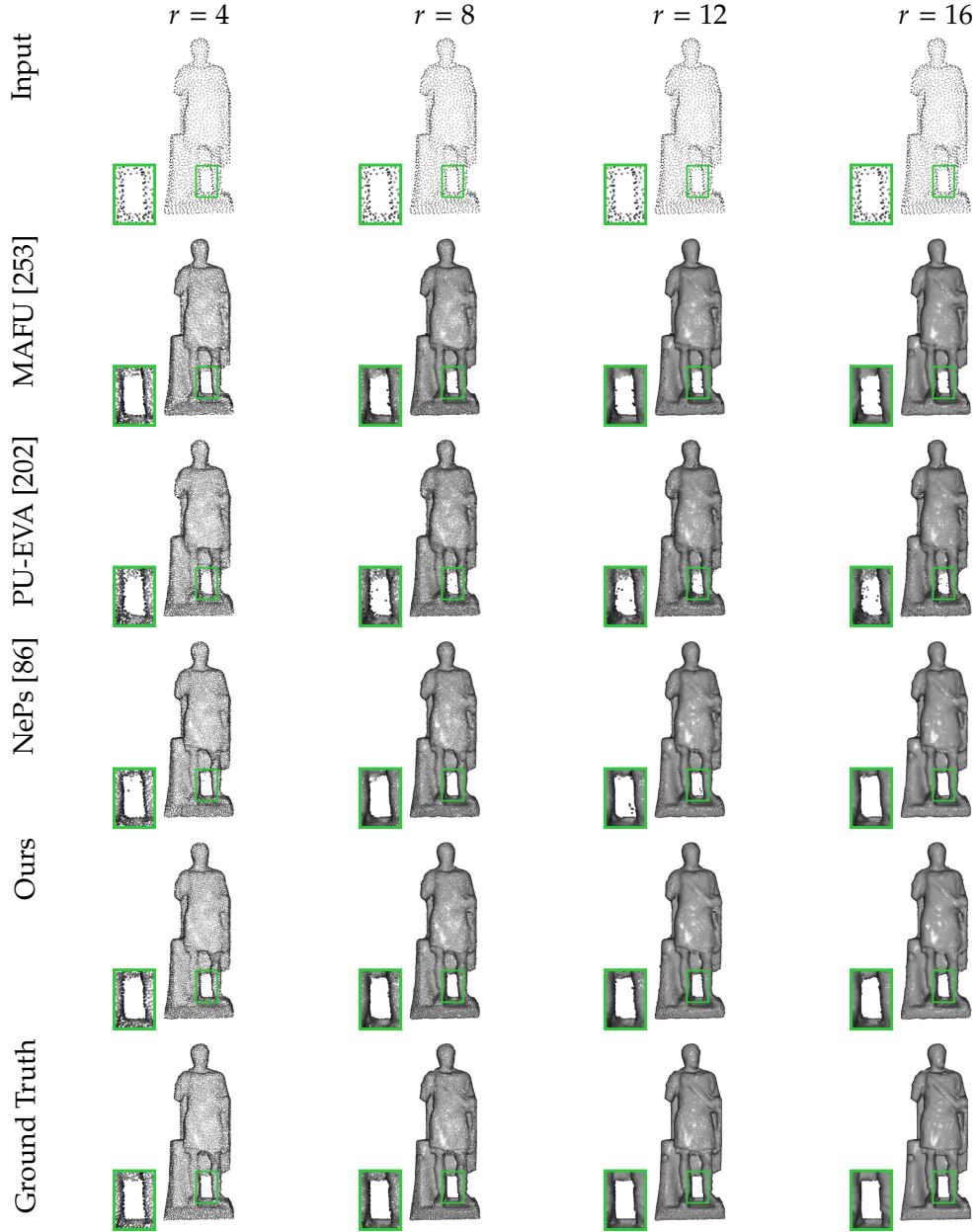


Figure 3.14: Qualitative comparison with state-of-the-art methods on the PU-GAN dataset and flexible upsampling ratios. Inputs with 2048 points are upsampled with $r \in \{4, 8, 12, 16\}$. Details are best viewed when zoomed in.

3.7. DISCUSSION

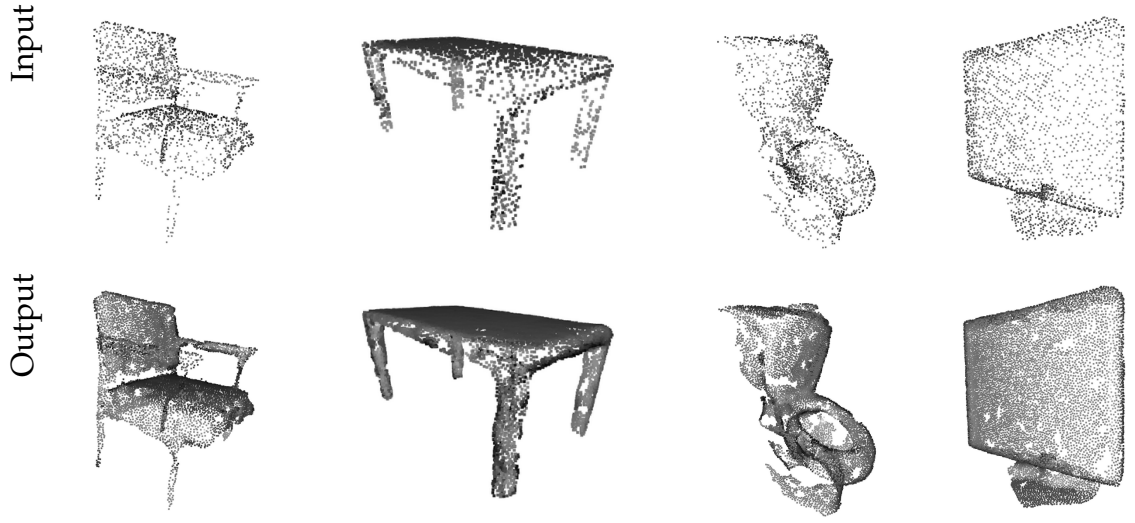


Figure 3.15: Qualitative upsampling results on the real-world ScanObjectNN dataset [312].

mapped to an intermediate representation as a Spherical Mixture of Gaussians; (ii) then, such distribution is sampled arbitrarily and the Transformer decoder learns to map each sample back to the surface. The predictions are further improved by an attention-based residual refinement module, which allows to achieve state-of-the-art results on different benchmarks. This strategy enables arbitrary upsampling, since the model is trained a single time with a fixed ratio and it can be queried at test time with any desired value.

4

Knowledge Distillation for Model Efficiency

The contributions of this chapter have been published in:

Dell'Eva, A., Orsingher, M., Lee, L., Bertozzi, M. (2023). Teacher Features-Driven Regularization for Knowledge Distillation. Under review.

4.1 STUDY OBJECTIVE AND CONTRIBUTIONS

The widespread diffusion of mobile and wearable devices requires efficient network inference on hardware with limited resources. However, modern deep learning models benefit from large architectures [113, 75, 314, 196, 299] that fail to meet the computational and memory requirement of such devices. In order to bridge this gap, three main neural compression approaches have been proposed: distilling the knowledge from large networks to smaller models [118, 265, 345, 369], pruning redundant parameters [83, 108, 273, 352, 187] and quantizing the weights to low-precision formats [93].

Knowledge Distillation (KD) is a model compression technique in which a lightweight *student* network is trained to mimic the outputs [118, 369, 214] or the intermediate representations [265, 345, 124, 304] of a much larger *teacher* model. This allows the knowledge learned by the teacher to be effectively transferred to the student, thus achieving a higher accuracy than training it from scratch.

Existing KD methods fall into two main categories: logit-based [118, 369,

4.1. STUDY OBJECTIVE AND CONTRIBUTIONS

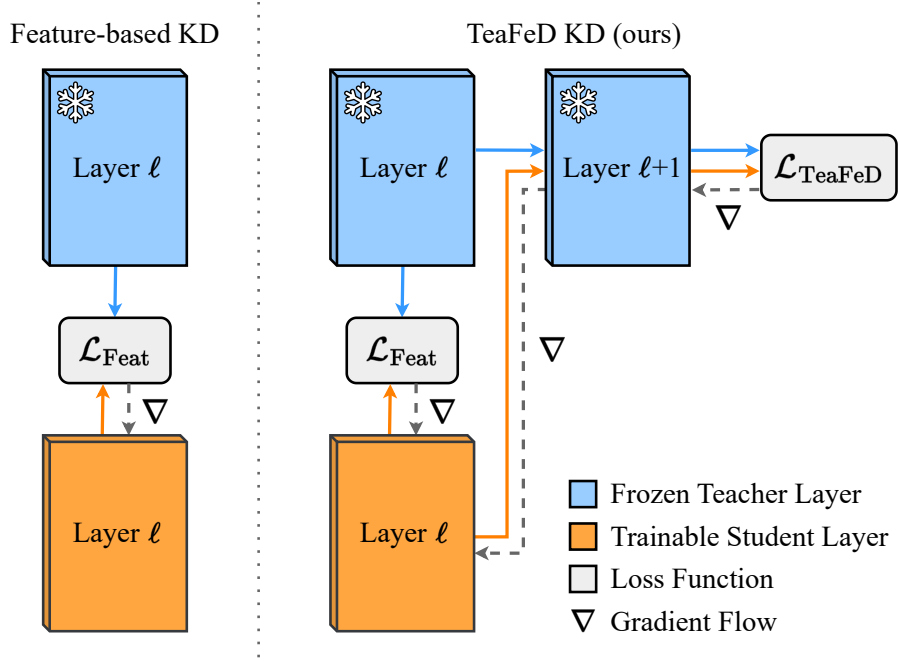


Figure 4.1: Overview of the proposed regularization loss for KD. We inject the student features at layer ℓ to the next *frozen* teacher layer $\ell + 1$ and match them with the corresponding teacher features.

214] and feature-based [265, 345, 124, 304]. KD was originally proposed in [118] for the classification task, as a method to match the output logits of two models with different capacity. The logits are converted to pseudo-probabilities with a softmax with rescaled temperature, and the KL divergence between the two distributions is minimized. Recently, feature-based methods have emerged by following the insights in [265]. The key idea is to align the intermediate representations of the student to those of the teacher, in order to boost the distillation performances.

However, directly matching the feature maps of the teacher is not a sufficient condition for the student to predict the correct output. Moreover, it is a hard optimization constraint to satisfy when the two models have different architectures or capacity. To this end, we propose *TeaFeD*, a novel *Teacher Features-Driven* regularization approach for KD, which is visualized in Fig. 4.1. The key idea is to train a generic student layer ℓ to produce a latent representation for the next teacher layer $\ell + 1$ that matches the corresponding teacher features. Since network layers typically implement non-injective functions, different features at level ℓ might produce the same output at level $\ell + 1$. Our approach allows for gradients to be propagated from a higher level of abstraction, thus providing

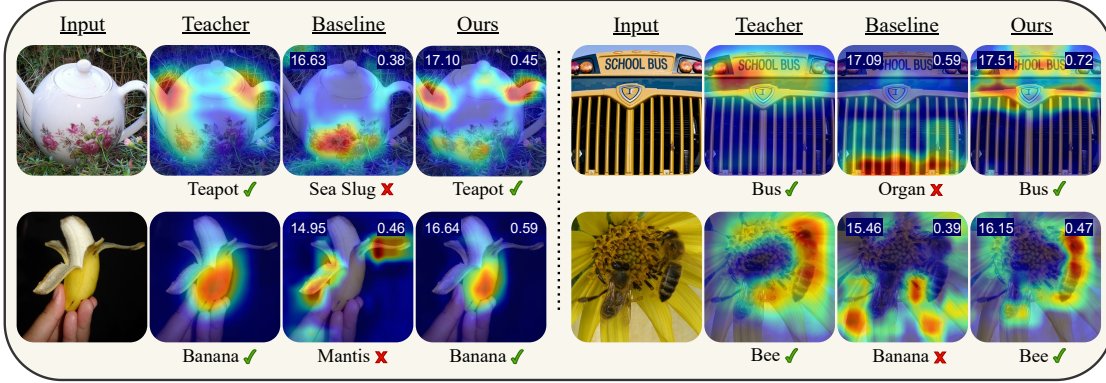


Figure 4.2: Visualization of the representations learned by the student with the baseline method [265] and our approach. The Euclidean and cosine distances, shown in the upper left and right corners of the students results, respectively, confirm that the baseline produces closer features to the teacher. However, it fails to classify inputs correctly as it focuses on irrelevant pixels, while our approach matches the teacher’s attention. Activation maps are computed with Grad-CAM [277], while distances are computed on the last backbone feature vector.

more flexibility to the student in learning intermediate representations.

The effect of this regularization loss on the features learned by the student network is shown in Fig. 3.3. Even if the direct matching baseline [265] produces closer representations to the teacher, as determined by both Euclidean and cosine distances, it fails to classify correctly the input images. The activation maps, computed with Grad-CAM [277], prove that these errors stem from the focus on irrelevant pixels, while our approach allows the student to concentrate on the same image regions as the teacher. For example, the colorful teapot decoration and the school bus radiator grille are misinterpreted by the baseline as sea slug and organ, respectively, while our method yields the correct output attending to the relevant image region.

Therefore, our contributions to existing distillation literature can be summarized as follows:

- We propose a novel approach for feature-based KD that complements direct matching with a teacher features-driven regularization loss.
- We show that relaxing the feature matching constraint at corresponding layers by distilling knowledge from a higher level of abstraction enables the student model to learn more robust and generalizable latent representations.

4.2. RELATED WORK

- We prove the effectiveness of our regularizer on standard benchmarks, showing state-of-the-art results for a wide variety of vision tasks, including image classification, semantic/instance segmentation and object detection.

Furthermore, we present a novel neural compression strategy, called *sparse self-distillation*, that combines network pruning with KD. After the pruning process, the pruned model is typically fine-tuned for a few epochs, in order to partially retrieve the accuracy lost due to the removed parameters. However, in practice, there is a tradeoff between the desired reduction rate and the bound on accuracy loss. To this end, we propose to integrate KD in the pruning and fine-tuning loop, by setting the original unpruned model as the teacher. In this way, the accuracy gap of the student (i.e. the pruned network) can be significantly reduced, and model compression can be pushed further. While this approach works across different network architectures, pruning strategies and distillation methods, we show that the proposed TeaFeD regularization is particularly effective also in this setting.

4.2 RELATED WORK

Logit-based KD. Logit-based methods distill the knowledge only at output level, by minimizing the divergence between the logits predicted by the teacher and the student model [118, 369, 214, 366, 134]. The seminal KD paper [118] proposed to rescale the temperature parameter in the last softmax layer, in order to smooth the output distribution and focus on negative logits [106]. Recently, DKD [369] showed that reformulating the distillation loss into a weighted sum for positive and negative logits effectively decouples their contributions and further boosts performances. On the other hand, DML [366] employed a mutual learning paradigm to train both models simultaneously, while TAKD [214] introduced an intermediate teacher assistant to reduce the capacity gap between the teacher and the student. Finally, Lin *et al.* [134] presented a multi-level framework to distill logits across instances, batches and categories. Logit-based methods are efficient in terms of both memory and compute, but they typically achieve lower results with respect to state-of-the-art feature-based approaches.

Feature-based KD. Feature-based methods aim at transferring intermediate representations from the teacher to the student, in order to provide a richer

supervision [265, 345, 239, 340, 38, 304, 109, 344, 288, 193, 124, 32, 116, 361, 306, 39]. Following the idea presented in [265] to mimic the teacher features via a simple L_2 loss, several approaches have been developed to improve different aspects of the pipeline. Some works [124, 345, 344] proposed to mask the feature maps, thus allowing the student to focus on relevant parts of the image. Popular masking strategies include learnable tokens [124], random masks reconstructed with generative layers [345], and fixed separation of background and foreground for object detection [344]. Another approach to alleviate the constraint of direct feature matching is to transform the raw features into attention maps [361, 109, 344, 288] or to compute local statistics, such as pairwise similarities [306] and the Pearson correlation coefficient [32]. Furthermore, cross-level distillation has been explored in [239, 38], with the goal of sharing features across different network layers, thus integrating knowledge from multiple learning stages. Inspired by these methods, we propose a straightforward approach to complement direct feature matching with a novel regularization term that forces each student layer to learn meaningful features from the subsequent teacher layers.

Network Pruning. The goal of network pruning is to remove redundant parameters from a given model, thus making it smaller and faster for inference. *Unstructured* pruning approaches [108, 273, 234, 169, 74] directly zero out individual weights based on their magnitude, without altering the network architecture. Despite being a generic and straightforward strategy, unstructured pruning typically requires dedicated hardware that supports sparse matrix multiplication. On the other hand, *structured* pruning methods [83, 352, 187, 172, 73] physically remove entire channels or layers from the network and allow to exploit standard GPU acceleration. In both cases, the pruning ratio is practically limited by the accuracy lost due to the removed parameters, even after fine-tuning the remaining ones. We propose to push further this sparsity limit by self-distilling the dense (unpruned) model to the sparse (pruned) network, during the pruning and fine-tuning loop.

4.3 METHOD

4.3.1 PRELIMINARIES

Logit-based KD. The fundamental baseline in KD research is the vanilla logit-based method introduced in [118]. Given a model trained on a classification task, the output logits $\mathbf{z} \in \mathbb{R}^C$ can be converted to pseudo-probabilities using the softmax activation as:

$$y_j = \frac{e^{z_j/t}}{\sum_{c=1}^C e^{z_c/t}}, \quad (4.1)$$

where z_j and y_j are the logit and probability values for j -th category, respectively. In the KD context, the *temperature* parameter t is often larger than 1, producing softer distributions which alleviate the over-confidence problem in neural networks [106] ($t = 1$ yields the vanilla softmax output). To transfer knowledge from complex to compact models, the original logit-based technique minimizes the Kullback-Leibler (KL) divergence between teacher and student outputs as:

$$\mathcal{L}_{\text{Log}} = \text{KL}(\mathbf{y}^T \parallel \mathbf{y}^S) = \sum_{j=1}^C y_j^T \log \left(\frac{y_j^T}{y_j^S} \right), \quad (4.2)$$

where the superscripts T and S indicate whether the outputs originate from the teacher or the student, respectively. As a result, the *dark knowledge* is transferred via soft labels from the teacher, improving the prediction of the student.

Feature-based KD. The distillation approaches based on features are versatile and mostly agnostic with respect to the model architecture, which can vary considerably for different tasks. The basic feature-based method [265] can be formulated as:

$$\mathcal{L}_{\text{Feat}}^\ell = \|F_\ell^T - \mathcal{A}_\ell(F_\ell^S)\|_2^2. \quad (4.3)$$

Here, $F_\ell^T \in \mathbb{R}^{C_{T_\ell} \times H_{T_\ell} \times W_{T_\ell}}$ and $F_\ell^S \in \mathbb{R}^{C_{S_\ell} \times H_{S_\ell} \times W_{S_\ell}}$ correspond to the teacher and student feature maps extracted from layer ℓ , while $\mathcal{A}_\ell$ denotes the convolutional adaptation layer needed to align the student features dimensions with those of the teacher.

The approach forces the student to directly imitate the teacher features. This may not be optimal if the models' architectures diverge significantly or their capacities are widely different, given the student's inability to exactly

replicate the more complex features. To address the issue, we introduce a novel feature-based regularization, which employs a straightforward L_2 loss to steer the student toward a more robust solution space.

4.3.2 TEACHER FEATURES-DRIVEN REGULARIZATION

Given a teacher network with a structured hierarchy of layers (or stages) denoted as $\{\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_L\}$, and a parallel student network denoted as $\{\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_L\}$, our TeaFeD method regularizes the learning process at matching layers. At each stage ℓ , the aligned student features $\mathcal{A}_\ell(F_\ell^S)$ are propagated to the corresponding subsequent frozen teacher stage $\mathcal{T}_{\ell+1}$, which provides new features on a higher level of abstraction. Thus, the loss is formulated as:

$$\mathcal{L}_{\text{TeaFeD}}^\ell = \|F_{\ell+1}^T - \mathcal{T}_{\ell+1}(\mathcal{A}_\ell(F_\ell^S))\|_2^2. \quad (4.4)$$

Due to non-linear mappings and pooling transformations, the teacher layer implementing the loss function is typically non-injective. This suggests that different inputs might produce identical latent representations, allowing for more flexibility in the features derived by the student. Therefore, our penalization term effectively acts as a powerful regularizer, promoting better generalization.

Overall loss. We employ a simple exponentially increasing weighting scheme on the sequence of features for both our loss and the vanilla feature-based loss. The aggregated losses can be expressed as, respectively:

$$\mathcal{L}_{\text{TeaFeD}} = \sum_{\ell=1}^L \gamma_1^{-\ell} \mathcal{L}_{\text{TeaFeD}}^\ell, \quad (4.5)$$

$$\mathcal{L}_{\text{Feat}} = \sum_{\ell=1}^L \gamma_2^{-\ell} \mathcal{L}_{\text{Feat}}^\ell. \quad (4.6)$$

Student models are trained with the following cumulative loss:

$$\mathcal{L} = \mathcal{L}_{\text{Task}} + \mathcal{L}_{\text{Log}} + \alpha \mathcal{L}_{\text{Feat}} + \beta \mathcal{L}_{\text{TeaFeD}}, \quad (4.7)$$

where α and β are hyperparameters to balance the losses and $\mathcal{L}_{\text{Task}}$ represents the task-specific loss (e.g., for classification, the cross-entropy computed between student outputs and ground truth labels). In practice, we set $\alpha = \beta$ and $\gamma_1 =$

4.3. METHOD

$\gamma_2 = \gamma \in (0, 1]$ to halve the number of hyperparameters to be tuned. The core steps of our approach are delineated in Algorithm 1.

Algorithm 1 TeaFeD Regularization in PyTorch

```

# (img, y): Input and Label
# (S, T): Student and Teacher
# num_layers: Number of Layers
# [adapt]: Adaptation Layers
# (g1, g2, a, b): Loss Hyperparameters
import torch.nn.functional as F
y_S, feat_S = S(img)
y_T, feat_T = T(img)
L_task = F.cross_entropy(y_S, y)
L_log = F.kl_div(y_S, y_T)
L_feat = 0.
L_teafed = 0.
for l in range(num_layers):
    F_S = adapt[l](feat_S[l])
    l2_feat = F.mse_loss(F_S, feat_T[l])
    L_feat += (g2 ** -1) * l2_feat
    l2_teafed = F.mse_loss(T[l+1](F_S), feat_T[l+1])
    L_teafed += (g1 ** -1) * l2_teafed
L = L_task + L_log + a * L_feat + b * L_teafed

```

4.3.3 SPARSE SELF-DISTILLATION

The proposed regularization loss allows to train an efficient student model with comparable accuracy with respect to the teacher network. To further enhance the deployment performances of the student, we also present a novel compression strategy, called *sparse self-distillation*, which integrates the distillation procedure in the pruning and fine-tuning loop.

Let S be the student network after KD training, $\mathcal{P}_r$ a generic pruning algorithm [83, 108, 273] with sparsity r and $\hat{S}_r = \mathcal{P}_r(S)$ the resulting model when applying $\mathcal{P}_r$ to S . Typically, $\hat{S}_r$ is fine-tuned for a few epochs with ground truth data and the task-specific loss $\mathcal{L}_{\text{Task}}(\hat{S}_r)$, in order to partially retrieve the accuracy lost during pruning. However, in practice, there is a tradeoff between the target sparsity and the bound on accuracy loss, especially for small networks already compressed with KD.

We propose to alleviate this constraint by setting the original unpruned student S as the *teacher* and to self-distill it during the pruning process. This

means that $\hat{S}_r$ is fine-tuned with the sum of two loss terms:

$$\mathcal{L} = \mathcal{L}_{\text{Task}}(\hat{S}_r) + \mathcal{L}_{\text{KD}}(\hat{S}_r, S) \quad (4.8)$$

The distillation loss $\mathcal{L}_{\text{KD}}$ forces the sparse student $\hat{S}_r$ to follow the output distribution and the learned representations of the original dense student S . In this way, the sparsity r can be pushed further, leading to the deployment of smaller and faster models with competitive accuracy. Note that this strategy is agnostic to the distillation method implemented by $\mathcal{L}_{\text{KD}}$. However, we show in Sec. 4.4.6 that our regularization loss is particularly effective in this setting, for both structured and unstructured pruning algorithms.

4.4 EXPERIMENTS

4.4.1 CLASSIFICATION

Datasets and baselines. We employ two standard image classification datasets for our experiments. CIFAR-100 [153] consists of 60k images (50k for training and 10k for validation) from 100 classes, with an image resolution of 32×32 . TinyImageNet [164] is a subset of ImageNet [67] and comprises 90k images from 200 categories, divided into 80k for training and 10k for validation, with resolution 64×64 . In addition to the basic KD method [118], we reproduce various state-of-the-art strategies for comparison, including FitNet [265], CRD [304], SRRL[340], SemCKD [38], ReviewKD [239], and CAT-KD [109]. Note that all the approaches, except for KD itself, incorporate the vanilla KD loss (Eq. 4.2).

Training settings. Following [113, 362], we employ standard data augmentation and normalize all images based on channel means and standard deviations. For CIFAR-100, the total training epochs are 240, the batch size is set to 64, and the base learning rate for SGD optimizer to 0.05. The hyperparameters for our loss are configured as $\{\alpha = 20, \gamma = 0.8\}$. For TinyImageNet, the models are trained for 200 epochs, the batch size is 128 and the base learning rate is 0.1, while the settings for the loss parameters are $\{\alpha = 5, \gamma = 0.8\}$. For classification, distillation is performed on the backbone features. The ablation study on losses and features employed can be found in Sec. 4.4.4.

4.4. EXPERIMENTS

Architecture	Homogeneous				Heterogeneous			
Student	ResNet-116	VGG-8	ResNet-8x4	WRN-40-1	VGG-8	ShuffleNetV1	MobileNetV2	ShuffleNetV2
	73.70	71.00	73.30	72.17	71.00	71.15	65.94	73.33
KD [118]	76.26	73.43	74.66	74.12	72.51	75.27	69.35	75.84
FitNet [265]	76.48	73.54	74.46	73.86	72.82	75.36	69.49	75.91
CRD [304]	76.83	74.37	75.72	74.44	73.80	75.88	70.05	76.40
SRRL [340]	76.79	73.85	75.51	74.78	73.49	75.59	69.96	76.19
SemCKD [38]	77.07	74.35	76.00	74.39	75.33	75.78	70.26	77.97
CAT-KD [109]	77.39	74.46	76.78	75.10	75.66	75.43	69.84	77.01
Ours	77.54	75.01	76.64	75.18	75.74	76.36	70.94	78.84
Teacher	ResNet-110x2	VGG-13	ResNet-32x4	WRN-40-2	ResNet-32x4	VGG-13	WRN-40-2	ResNet-32x4
	78.17	75.14	79.21	76.53	79.21	75.14	76.53	79.21

Table 4.1: Top-1 test accuracy (%) of homogeneous and heterogeneous teacher-student pairs on CIFAR-100. Average over 4 runs.

Architecture	Homogeneous		Heterogeneous	
Student	VGG-8	ResNet-8x4	ShuffleNetV1	VGG-8
	55.83	55.53	58.64	55.83
KD [118]	60.36	58.16	61.25	60.52
FitNet [265]	60.35	58.21	61.47	60.69
CRD [304]	61.72	59.62	63.50	61.81
SRRL [340]	61.41	59.72	62.74	62.02
SemCKD [38]	61.41	57.50	62.53	62.57
CAT-KD [109]	61.52	58.13	63.70	62.59
Ours	62.24	60.28	64.21	63.75
Teacher	VGG-13	ResNet-32x4	VGG-13	ResNet-32x4
	61.34	64.53	61.34	64.53

Table 4.2: Top-1 test accuracy (%) of homogeneous and heterogeneous teacher-student pairs on TinyImageNet. The results are averaged over 3 runs.

Results. The classification results are shown in Tab. 4.1 and Tab. 4.2, with evaluations based on Top-1 accuracy. It can be noticed that we consistently outperform other state-of-the-art approaches for nearly every pair on both datasets, including ReviewKD [239] and CAT-KD [109] which require careful hyperparameter tuning for each teacher-student combination. Remarkably, we improve accuracy by a considerable margin in several cases. For example, with the ShuffleNetV2/ResNet-32x4 pair on CIFAR-100, we achieve a 1% accuracy increase compared to the second-best method. Similarly, in the VGG-8/VGG-13 combination on TinyImageNet, the student surprisingly exceeds even the teacher performance. The outcomes prove that the introduced regularization term facilitates flexible feature learning, enhancing the student robustness regardless of dataset and architecture types.

4.4.2 OBJECT DETECTION AND INSTANCE SEGMENTATION

Dataset and baselines. The experiments are conducted on COCO2017 dataset [185], which comprises 120k training images and 5k validation images divided in 80 object categories. We compare our approach with recent distillation methods for detectors, including FKD [283], CWD [288], FGD [344], MGD [345], and MaskD [124].

Training settings. We train the models for 24 epochs with SGD optimizer (momentum = 0.9, weight decay = 10^{-4}). When the student and teacher share the same head structure, we adopt the inheriting strategy [344, 345] to initialize the student with the teacher’s neck and head parameters. The hyperparameters are $\{\alpha = 5 \times 10^{-5}, \gamma = 1\}$ for one-stage detectors and $\{\alpha = 5 \times 10^{-7}, \gamma = 1\}$ for two-stage detectors. The distillation loss is computed on the feature maps extracted from the neck. The experiments are executed with MMDetection [41].

Results. The experiments for the object detection task include three different models, namely the one-stage detector RetinaNet [184], the anchor-free one-stage detector RepPoints [343] and the two-stage detector Faster RCNN [261]. The evaluation metrics are the mean bounding box Average Precision (mAP) and the variants AP_S , AP_M , and AP_L , corresponding to small, medium and large object sizes, respectively. In the context of instance segmentation, we focus on SOLO [323] and Mask RCNN [114] models, which are evaluated with the mean mask Average Precision metric (mAP). From the results presented in Tab. 4.3 and Tab. 4.4, it can be noticed that our method achieves better performances with respect to the other state-of-the-art approaches for both tasks. For instance, our TeaFeD regularization boosts significantly the RepPoints and Mask RCNN students, leading to 4.3 and 2.7 mAP improvement, respectively. Additionally, RetinaNet and SOLO obtain a remarkable 2.8 and 3.1 improvement on the AP_S metric.

4.4.3 SEMANTIC SEGMENTATION

Dataset and baselines. Semantic segmentation is evaluated on Cityscapes dataset [59], consisting of 5k urban scenes images divided in 3k for training, 500 for validation and 1.5k for testing. We compare our method with recent

4.4. EXPERIMENTS

	Teacher	Student	mAP	AP _S	AP _M	AP _L
One-stage	RetinaNet ResNeXt101 41.0	RetinaNet-Res50	37.4	20.6	40.7	49.7
		FKD [283]	39.6	22.7	43.3	52.5
		CWD [288]	40.8	22.7	44.5	55.3
		FGD [344]	40.7	22.9	45.0	54.7
		MGD [345]	41.0	23.4	45.3	55.7
		MasKD [124]	41.0	22.6	45.3	55.3
		Ours	41.3	23.4	45.9	54.6
Anchor-free	RepPoints ResNeXt101 44.2	RepPoints-Res50	38.6	22.5	42.2	50.4
		FKD [283]	40.6	23.4	44.6	53.0
		CWD [288]	42.0	24.1	46.1	55.0
		FGD [344]	42.0	24.0	45.7	55.6
		MGD [345]	42.3	24.4	46.2	55.9
		MasKD [124]	42.5	24.9	46.1	56.8
		Ours	42.9	24.8	47.1	56.3
Two-stage	Cascade Mask RCNN ResNeXt101 47.3	Faster RCNN-Res50	38.4	21.5	42.1	50.3
		FKD [283]	41.5	23.5	45.0	55.3
		CWD [288]	41.7	23.3	45.5	55.5
		FGD [344]	42.0	23.8	46.4	55.5
		MGD [345]	42.1	23.7	46.4	56.1
		MasKD [124]	42.4	24.2	46.7	55.9
		Ours	42.4	24.4	46.3	56.4

Table 4.3: Object detection results on COCO.

	Teacher	Student	mAP	AP _S	AP _M	AP _L
One-stage	SOLO-Res101 3x, MS 37.1	SOLO-Res50-1x	33.1	12.2	36.1	50.8
		FGD [344]	36.0	14.5	39.5	54.5
		MGD [345]	36.2	14.2	39.7	55.3
		Ours	36.4	15.3	40.1	55.2
Two-stage	Cascade Mask RCNN ResNeXt101 41.1	Mask RCNN-Res50	35.4	16.6	38.2	52.5
		FGD [344]	37.8	17.1	40.7	56.0
		MGD [345]	38.1	17.1	41.1	56.3
		Ours	38.1	17.8	41.0	56.5

Table 4.4: Instance segmentation results on COCO.

Architecture	Homogeneous	Heterogeneous
Student	PspNet-Res18 69.85	DeepLabV3-Res18 73.20
SKDS [193]	72.70	73.87
CWD [288]	73.53	75.93
MGD [345]	74.10	76.31
Ours	74.87	76.44
Teacher	PspNet-Res101 78.34	PspNet-Res101 78.34

Table 4.5: Semantic segmentation results on Cityscapes.

approaches, such as SKDS [193], CWD [288] and MGD [345].

Training settings. We train the student models with SGD (momentum = 0.9, weight decay = 5×10^{-4}) for 128 epochs with batch size equal to 16. Following [345], our loss is computed only on the last backbone feature and the corresponding parameters are set to $\{\alpha = 2 \times 10^{-5}, \gamma = 1\}$. The experiments are conducted with MMSegmentation [57].

Results. We evaluate the segmentation task on both homogeneous and heterogeneous distillation settings, using mean Intersection over Union (mIoU) as the quantitative metric. Specifically, we employ PspNet-Res101 [373] as the teacher, while PspNet-Res18 [373] and DeepLabV3-Res18 [46] as students. The results in Tab. 4.5 confirm that our method is the best among the state-of-the-art approaches. In detail, the two students register a 5.02 and 3.24 improvement with respect to the original model trained without distillation, respectively.

4.4.4 ABLATIONS

We conduct several ablation studies to assess the contribution of distinct layers and losses on the model predictions. The results in Tab. 4.6 are obtained by training a homogeneous and a heterogeneous classification network pair on CIFAR-100 with different configurations.

In order to evaluate the influence of layers/blocks, we perform distillation removing them sequentially. Note that, in this context, layers are enumerated from highest to lowest, with $\ell = 1$ corresponding to the last layer of the backbone.

4.4. EXPERIMENTS

Architecture	Homogeneous	Heterogeneous
Student	VGG-8 71.00	ShuffleNetV2 73.33
$\ell \in \{1, 2, 3\}$	74.55	78.79
$\ell \in \{1, 2\}$	74.49	78.64
$\ell = 4$	73.41	76.74
$\ell = 3$	73.53	76.12
$\ell = 2$	74.11	77.81
$\ell = 1$	74.21	77.27
w/o $\mathcal{L}_{\text{Log}}$	74.53	78.73
w/o $\mathcal{L}_{\text{Feat}}$	74.70	78.09
w/o $\mathcal{L}_{\text{TeaFeD}}$	74.49	78.07
Ours	75.01	78.84
Teacher	VGG-13 75.14	ResNet-32x4 79.21

Table 4.6: Ablation studies on different versions of the student for both homogeneous and heterogeneous settings. Layers are numbered in descending order. Ours is equivalent to $\ell \in \{1, 2, 3, 4\}$. Results are averaged over 3 runs.

Two observations emerge from the accuracy values: firstly, integrating more layers is beneficial for the student, as demonstrated by training the model with only a limited subset; secondly, the final layers contribute more to the network performance, as highlighted by the improved numerical evaluations at $\ell = 1$ and $\ell = 2$. This justifies the choice of applying exponential weighting to the feature-based losses, emphasizing the terminal layers.

Moreover, we investigate the impact of the different losses on the student effectiveness. The last rows of Tab. 4.6 confirm that our regularization objective plays a crucial role in the distillation process. Notably, removing our term leads to the most significant performance drop compared to the exclusion of the other distillation losses. Additionally, the results show that, when evaluated independently, our loss outperforms the vanilla feature-based approach. Nonetheless, the best accuracy is achieved through joint training.

4.4.5 ANALYSIS

In this section, we provide an extensive analysis of our KD approach in terms data and training efficiency, transferability of the features learned by the

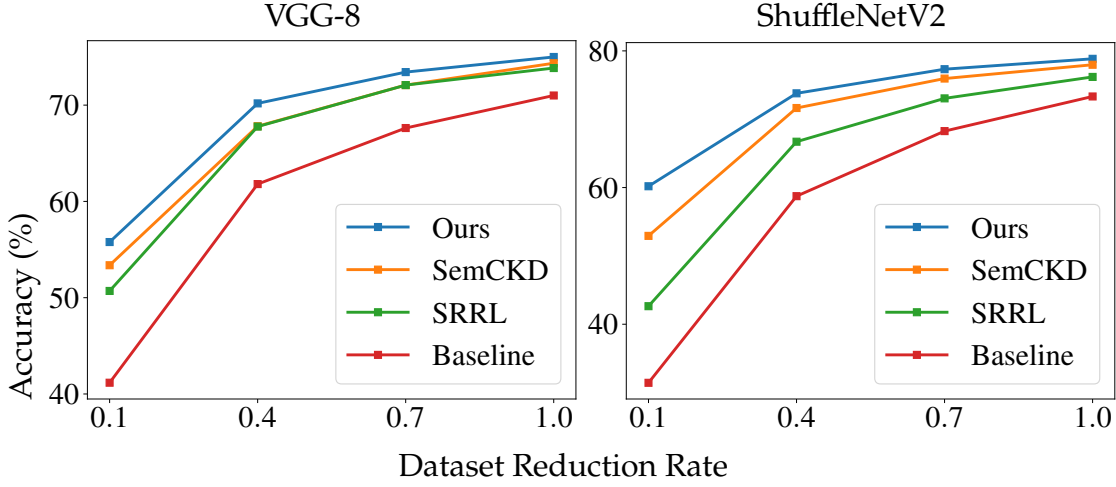


Figure 4.3: Analysis on the data efficiency of different KD methods when trained with different amounts of data on CIFAR-100 classification.

student, sensitivity to hyperparameters and impact of the adaptation layer. All the experiments are conducted for the classification task on CIFAR-100 [153] for a ShuffleNetV2 student distilled from a ResNet-32x4 teacher, except where stated otherwise.

Data Efficiency. We evaluate the dependency of different distillation approaches [38, 340] on the amount of training data, by reducing the training set of CIFAR-100 [153] to various ratios. Fig. 4.3 shows that we consistently outperform other methods across the whole spectrum of dataset reductions, for both ShuffleNetV2 and VGG-8 (with VGG-13 as teacher) architectures. This result proves that the proposed regularization loss can transfer knowledge from the teacher more efficiently.

Training Efficiency. While adding our $\mathcal{L}_{\text{TeaFeD}}$ as a learning objective leads to improved classification accuracy, it also increases the overall training time of the student network. We study this aspect by structurally pruning [83] the teacher at multiple sparsity values and analyzing the accuracy-efficiency tradeoff. Fig. 4.4 compares our results with the baseline feature matching approach [265]. It can be seen that distilling from a lighter teacher with our KD method allows to achieve approximately the same training time, while losing only 0.35% accuracy, which is almost 3× less than the baseline 0.98% accuracy loss. Moreover, all the intermediate sparsity levels provide a substantial efficiency gain with a

4.4. EXPERIMENTS

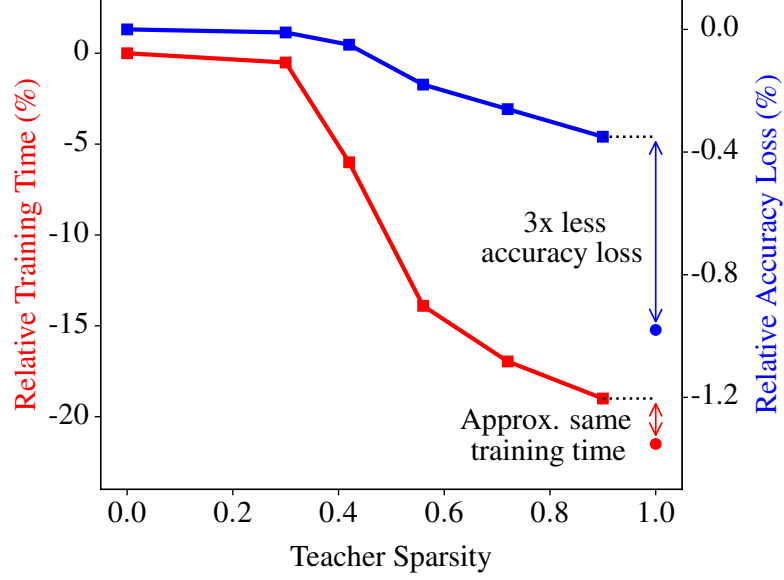


Figure 4.4: Analysis on how pruning the teacher allows to reduce the training time of our method almost to the baseline, while keeping much more accuracy.

bounded performance loss, allowing to adapt the method to custom computational constraints.

Features Transferability. As we are interested in which features are learned by the student during the distillation procedure, we also analyze how such features transfer to datasets unseen at training time. To this end, the student trained on CIFAR-100 [153] is used as a frozen representation extractor for images from STL-10 [55] and TinyImageNet [164]. Then, we train a linear classifier on these representations and evaluate the accuracy on both datasets. Tab. 4.7 shows the results against multiple baselines for homogeneous and heterogeneous teacher-student pairs. The regularization effect introduced by our proposed loss allows the student to learn more transferable features with respect to concurrent KD methods, thus proving its effectiveness.

Hyperparameters Sensitivity. The proposed approach depends on two main hyperparameters, namely the weights $\gamma \in (0, 1]$ of each feature layer and $\alpha \in \mathbb{R}$ for the feature-based KD losses. We show in Fig. 4.5 that the performances of our regularization term are minimally affected by the choice of such parameters. By letting $\gamma \in \{0.7, 0.8, 0.9\}$ and $\alpha \in \{5, 10, 20, 50, 100\}$, the accuracy difference between the best and worst configuration is less than 1%. Moreover, note we

Architecture	Homogeneous		Heterogeneous	
Teacher	ResNet-110x2		VGG-13	
Student	ResNet-116		ShuffleNetV1	
CIFAR-100 $\rightarrow$	STL-10	TinyImg	STL-10	TinyImg
KD [118]	63.67	29.59	67.70	36.00
FitNet [265]	64.22	30.64	67.94	36.30
SRRL [340]	64.22	31.59	69.15	37.58
SemCKD [38]	66.42	34.32	69.17	38.63
CRD [304]	66.11	32.69	69.03	38.66
CAT-KD [109]	65.73	32.75	70.82	39.36
Ours	66.68	34.11	70.86	39.52

Table 4.7: Features transferability from CIFAR-100 to other classification datasets. A linear classifier is trained on features extracted from a frozen student distilled with different KD methods.

avoid cherry-picking a specific set of good hyperparameters, as our choice in Sec. 4.4.1, i.e. $\{\alpha = 20, \gamma = 0.8\}$ is not even the optimal one.

Adaptation Layer. We compare several implementations of the adaptation layer $\mathcal{A}$, which aligns the student features with those of the teacher. The first rows of Tab. 4.8 suggest that any straightforward single-layer design is beneficial for the knowledge transfer procedure, providing comparable performances. Notably, the highest accuracy is achieved with a 3x3 convolutional layer followed by a batch normalization. This configuration is employed in every experiment across all tasks. Conversely, aligning the features with a three-layer transformation mitigates the distillation effectiveness. We conjecture this is due to the additional capacity provided by the multi-layer structure. In this case, the adaptation module can learn to transform any student representation into any complex teacher representation, preventing the student model from receiving useful feedback and thus learning robust intermediate features.

Method Extension. We extend our method by modifying the design of the loss. In this setting, the features at layer ℓ are propagated to all the corresponding subsequent layers of the teacher, i.e. until the final layer L . Thus, our regularization term is computed on the output logits using Eq. 4.2. As presented in Tab. 4.9, the accuracy values from this strategy are on par with our feature-driven loss,

4.4. EXPERIMENTS

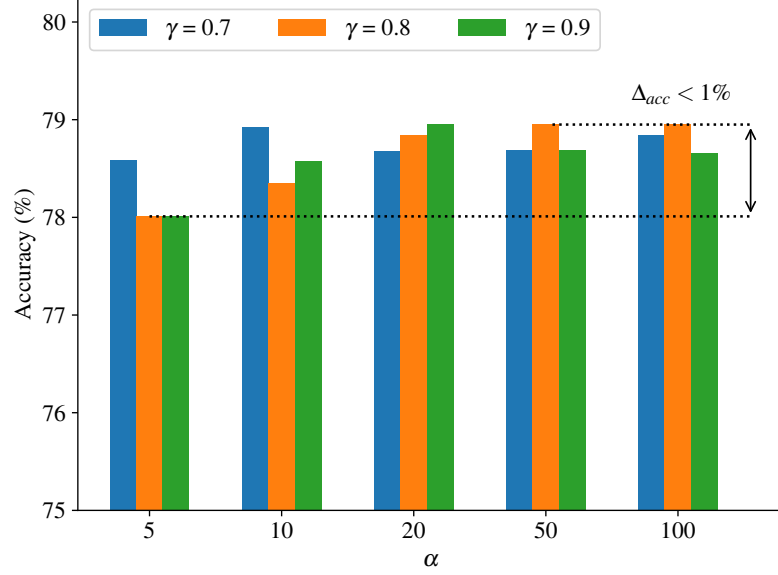


Figure 4.5: Analysis on the effect of varying the hyperparameters required to tune our proposed regularization loss.

Layer	Accuracy
Conv1x1-BN	78.75
Conv3x3-BN-NL	78.60
Conv1x1-Conv3x3-Conv1x1	78.34
Conv1x1-Conv3x3(DW)-Conv1x1	78.46
Ours (Conv3x3-BN)	78.84

Table 4.8: Comparison of adaptation layers. ConvNxN indicates a convolutional layer with a NxN kernel size, while BN, NL, and DW denote batch normalization, ReLU non-linearity, and depthwise convolution, respectively. Each convolution in the multi-layer configurations is followed by BN and NL.

	Student	Ours	Ours ⁺	Teacher
Homog.	ResNet-116	77.54	77.65	ResNet-110x2
	VGG-8	75.01	74.81	VGG-13
	ResNet-8x4	76.64	76.63	ResNet-32x4
	WRN-40-1	75.18	75.26	WRN-40-2
Heterog.	VGG-8	75.74	75.41	ResNet-32x4
	ShuffleNetV1	76.36	76.16	VGG-13
	MobileNetV2	70.94	70.77	WRN-40-2
	ShuffleNetV2	78.84	78.43	ResNet-32x4

Table 4.9: Top-1 test accuracy (%) of homogeneous and heterogeneous teacher-student pairs on CIFAR-100 obtained with our extended method, denoted as +.

Student	KD	DKD	Ours ⁺ w/ KD	Ours ⁺ w/ DKD	Teacher
ShuffleNetV2 73.33	75.84	77.02	78.43	78.70	ResNet-32x4 79.21

Table 4.10: Top-1 test accuracy (%) on CIFAR-100 obtained with our extended method using different logit-based approaches.

adding no significant advantages to the distillation process. Moreover, the primary limitation of this approach is the extended training time, which increases by 30% compared to our original configuration.

Additionally, in this revisited setup, we train the student with a different logit-based loss, applying DKD [369] to both $\mathcal{L}_{\text{Log}}$ and $\mathcal{L}_{\text{TeaFeD}}$. As expected, the adoption of a more effective training objective on the logits enhances the results, which can be found in Tab. 4.10. Despite the poor training efficiency, the setting can be leveraged as a pure logit-based distillation method (removing $\mathcal{L}_{\text{Feat}}$), boosting the student trained exclusively with $\mathcal{L}_{\text{Log}}$ when the features are not available. Furthermore, the configuration is versatile and adaptable to any logit-based technique.

4.4.6 SPARSE SELF-DISTILLATION

Settings. We test our sparse self-distillation procedure on the CIFAR-100 classification dataset [153], with different network architectures, including VGG-8, ResNet-116 and MobileNetV2, as well as various distillation algorithms [265, 340, 38]. We also compare the proposed method against the baseline approach

4.4. EXPERIMENTS

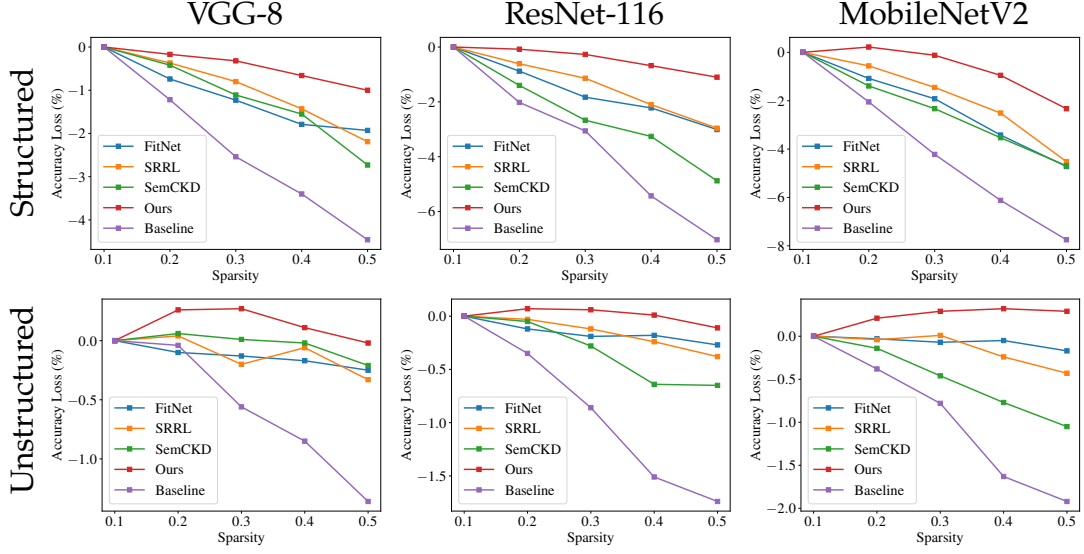


Figure 4.6: Quantitative results of our sparse self-distillation method for image classification on CIFAR-100. The baseline pruning and fine-tuning approach (without KD) is consistently outperformed by integrating KD in the loop for all networks, pruning and distillation algorithms. Moreover, our TeaFeD regularization loss provides the lowest accuracy loss across the whole sparsity spectrum.

of iterative pruning and fine-tuning on ground truth data, without KD. Unstructured pruning is applied with the built-in PyTorch utility, while we rely on the state-of-the-art DepGraph [83] algorithm for structured pruning. In all the experiments, the remaining parameters of the pruned network $\hat{S}_r$ are fine-tuned for 50 epochs.

Results. The sparse self-distillation results are shown in Fig. 4.6, where two key results emerge. Firstly, integrating KD within the pruning and fine-tuning loop is consistently better than the baseline approach, for any distillation method. Secondly, our proposed TeaFeD regularization is particularly effective in this setting, outperforming all the other KD losses. Such results find evidence across different architectures and pruning algorithms. This allows to push the network sparsity further, with a much smaller accuracy loss, thus leading to faster models for the inference stage.

4.5 DISCUSSION

In this work, we show that feature-based knowledge distillation can be improved by complementing the direct feature matching baseline with a teacher features-driven regularization loss, which enables the student model to learn robust intermediate representations. We extensively evaluate and ablate the proposed approach, outperforming state-of-the-art distillation methods on several vision tasks and benchmarks. Furthermore, we also introduce a novel neural compression strategy that effectively combines KD with network pruning, in order to improve the sparsity-accuracy tradeoff when deploying the student model on inference devices. The main limitation of our approach is the need for accessing the teacher weights at training time. As future work, we plan to tackle this limitation in order to support more general black-box teachers and to apply our method on non-convolutional architectures.



Conclusions

The contributions of the thesis have been published in:

Dell'Eva, A., Pizzati, F., Bertozzi, M., and de Charette, R. (2022). Leveraging local domains for image-to-image translation. In International Conference on Computer Vision Theory and Applications (VISAPP). *Best paper award*.

Dell'Eva, A., Orsingher, M., Bertozzi, M. (2022). Arbitrary Point Cloud Upsampling with Spherical Mixture of Gaussians. In International Conference of 3D Vision (3DV).

Dell'Eva, A., Orsingher, M., Lee, L., Bertozzi, M. (2023). Teacher Features-Driven Regularization for Knowledge Distillation. Under review.

In this dissertation, we systematically analyzed different methodologies to enhance deep learning models, introducing and implementing novel strategies to achieve this objective. Specifically, we focused on three main areas: the augmentation of training datasets, the refinement of input data during the inference stage, and the transfer of knowledge from complex to lightweight models via a process known as knowledge distillation.

In the context of the augmentation of training datasets, we proposed an image-to-image translation model composed by a GAN backbone and an optional VAE to interpolate between domains [65]. The system performs translations using only the source domain, thus hallucinating a new unseen domain, reducing the need for time-consuming labeling processes. Our few-shot

method showed superior performance in several tasks, such as lane degradation, snow addition and deblurring. Different real-world applications have been showcased, enhancing lane detection in degraded environments and improving semantic segmentation in snowy conditions. The study showed the use of generative models for data augmentation represents a valid alternative for training learning systems. Unlike traditional data augmentation techniques that rely on simple transformations like rotation or flipping, GANs can produce new synthetic data that are consistent with the patterns of the original dataset. By generating data that captures the underlying distribution of the real data, these models can considerably expand the size and diversity of training datasets. The enhanced dataset can have significant implications, mitigating overfitting, improving model robustness, and leading to better generalization in scenarios where data collection is challenging or limited.

Concerning the refinement of input data, we presented a Transformer-based architecture designed to decouple the upsampling process in two key steps, i.e. firstly mapping the sparse 3D input to a spherical mixture of gaussians and then arbitrarily sampling this distribution [64]. Such a flexible, efficient, and adaptable approach offers the versatility crucial for diverse applications. For example, we showed the improvement of reconstruction results derived by the refined input and the detailed upsampling of real-world urban driving scenes, which could be leveraged to enhance 3D object detection and segmentation. The exploration of the method effectiveness on these downstream tasks is planned for future studies. The research underlined the relevance of input quality during inference. When input data is of high quality and accurate, the model outputs are more likely to be precise and trustworthy. On the contrary, poor or noisy input can lead to incorrect predictions, even if the model has been trained on a vast and diverse dataset.

In the knowledge transfer field, we showed that directly matching latent representations is suboptimal, since network layers are typically non-injective functions and different features at an early level might produce the same output at later levels [66]. By augmenting the knowledge distillation process with a teacher features-driven regularization loss, the student learns more robust intermediate representations, leading to better performance across several vision tasks. However, the constraint of requiring access to teacher weights during training limits the flexibility of the approach. The introduced neural compression strategy, which combines knowledge distillation with network pruning,

offered promising results for deploying models efficiently on inference devices. Expanding the method to other model architectures will be addressed in future works. The study highlighted the significance of knowledge distillation, allowing the deployment of deep learning models on resource-constrained devices, such as mobile phones and edge devices, and concurrently ensuring that efficiency does not compromise accuracy.

In conclusion, our research showcased how a deep learning model can be used to enhance the performance of a different model. Implementing novel methods, we emphasized that leveraging the capabilities of a system can significantly improve the effectiveness and efficiency of another. These strategies highlight the potential benefits of collaborative learning, with models cooperating to achieve better results.



Overview of Deep Learning Technologies

A.1 GENERATIVE VS. DISCRIMINATIVE MODELS

Consider a set of data instances, denoted as X , and a corresponding set of labels, denoted as Y :

- *Generative models* aim to capture the joint probability distribution $p(X, Y)$. In the absence of labels, the model focuses on just $p(X)$. Their main objective is to understand and represent the underlying distribution of the data. For example, models designed to predict the next element in a sequence are primarily generative. GANs fall within this category, though many other model types exist, e.g. Hidden Markov Models (HMM) and Variational Autoencoders (VAE). A significant challenge for these models is capturing complex correlations, such as the relationship between contextual elements in images.
- *Discriminative models* are designed to determine the conditional probability $p(Y|X)$. They do not necessarily compute the probability of a data instance but rather focus on determining if a particular label can be assigned to a given instance. Their purpose is to differentiate and classify data based on certain patterns.

It is important to notice that not all models are inherently designed to produce a probability value. For instance, a discriminative classifier like a decision

A.2. ARCHITECTURES

tree might assign a label without specifying a probability. Nonetheless, it remains a model because the distribution of all predicted labels would model the real distribution of labels in the data. Similarly, generative models can model a distribution by producing synthetic data that resembles the authentic distribution.

From a comparative point of view, generative models tackle a more comprehensive and difficult task than their discriminative counterparts. For instance, a generative model for images would need to understand complex relationships, such as correlations within the image. Conversely, a discriminative model may be able to distinguish between categories by identifying a limited number of patterns, avoiding many details that a generative model must capture. In summary, while discriminative models try to draw boundaries in the data space, generative models try to model how data is placed throughout the space.

A.2 ARCHITECTURES

In this section, we provide a brief introduction to two relevant architectures in deep learning: the Convolutional Neural Network (CNN) and the Transformer. While other architectures exist in the field, our study focuses on these two due to their popularity and significance. Architectures beyond these are outside the scope of this overview.

A.2.1 CONVOLUTIONAL NEURAL NETWORKS

In deep learning, Convolutional Neural Networks (CNNs) are one of the most commonly employed models. The main advantage of CNNs over earlier algorithms is their capability to identify relevant features without the necessity for human intervention. Their versatility is showcased by extensive applications across different domains, including computer vision and speech processing. The architecture of CNNs draws inspiration from the neuronal structures found in both human and animal brains, similar to vanilla neural networks. Delving deeper into this analogy, the visual cortex in a cat’s brain consists of a complex sequence of cells, which is simulated by CNNs [126].

Goodfellow et al. [99] highlighted three key benefits of CNNs: equivariant representations, sparse interactions, and parameter sharing. CNNs are equivariant with respect to translation, meaning that translating a convolutional layer

input will result in a translation in the output. Moreover, contrary to conventional fully connected (FC) networks, CNNs employ shared weights and local connections. This design is particularly optimized to exploit the structure of 2D data, like images. As a result, fewer parameters are required, which simplifies the training process and boosts the network efficiency. Notably, these neurons focus on particular portions of a visual scene instead of its entirety. Essentially, they detect local patterns in the input, functioning similarly to local filters.

A common version of CNNs, which shares similarities with the Multi-Layer Perceptron (MLP), consists of several convolutional layers followed by sub-sampling or pooling layers. The final parts are composed of FC layers. In a CNN model, the input x for each layer is characterized by three dimensions: height, width, and depth, represented mathematically as $h \times w \times d$. Usually, the height is equivalent to the width, while the depth dimension is also referred to as the channel number. As an example, for an RGB image, the depth is equal to three. Within each convolutional layer, there are several kernels (or filters) denoted by k , which also have a three-dimensional structure, represented as $n \times n \times c$. Notably, n is strictly less than h , and c is less than or equal to d . The local connections allow the sharing of parameters, specifically the kernel weights W_k and the bias b_k . Through the convolution of these kernels with the input, k feature maps, denoted as h_k are produced. Next, introducing nonlinearity through an activation function f , such as ReLU, sigmoid and tanh, yields:

$$h_k = f(W_k * x + b_k). \quad (\text{A.1})$$

Following the generation of feature maps, each one undergoes down-sampling via the sub-sampling layers. For every feature map, a specific pooling function—such as max or average pooling—is applied over a continuous region sized $p \times p$, where p represents the kernel dimensions. Finally, the FC layers integrate the low and mid-level features to form high-level abstractions. The final layer is responsible for producing the task results. An illustration of a simple CNN architecture can be found in Fig. A.1.

Popular CNN architectures include LeNet [165], the first CNN in the image domain, AlexNet[154], the first breakthrough over traditional methods, VGG [289] and ResNet[113], which introduced deeper models and residual connections, DenseNet [119], that proposed densely connected layers, and GoogleNet [295], that presented the inception layer.

A.2. ARCHITECTURES

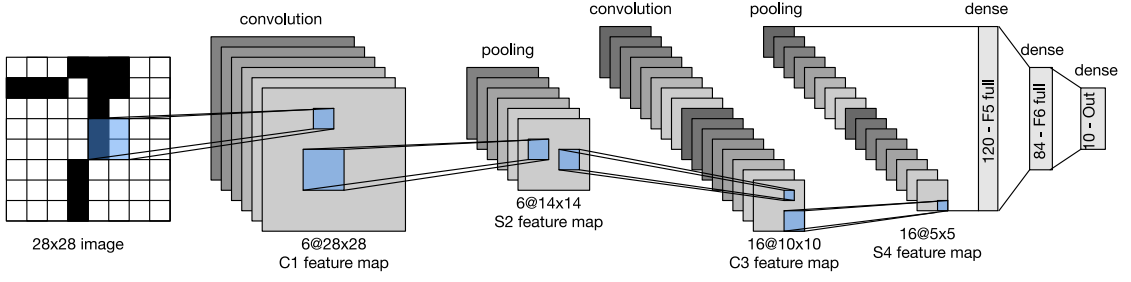


Figure A.1: The LeNet architecture [165]. The input is a handwritten digit, the output is a probability over 10 possible outcomes. Source: [363].

A.2.2 TRANSFORMERS

Currently, the predominant models for most natural language processing (NLP) tasks are based on the Transformer architecture. For any emergent task in NLP, the standard initial approach is to utilize a large pretrained Transformer-based model, such as BERT [69], RoBERTa [194], or Longformer [13]. These models are then fine-tuned based on the available data for the specific task after adapting the output layers accordingly. Recent developments surrounding large language models are the results of Transformer-based models such as GPT-2 and GPT-3 [255, 29]. In parallel, the vision Transformer has been established as a default model for a range of vision tasks such as image recognition, object detection, semantic segmentation, and superresolution [75, 195]. Additionally, Transformers have become relevant for other tasks like speech recognition [104], reinforcement learning [47], and graph neural networks [77].

The vanilla Transformer, as presented by Vaswani et al. [314], is a sequence-to-sequence model and consists of both an encoder and a decoder. Each of these components is constructed with L identical layers. Generally, every encoder layer is formulated using a multi-head self-attention mechanism coupled with a position-wise feed-forward network (FFN). To facilitate the development of deeper networks, a residual connection [113] is integrated around each module, which is subsequently followed by Layer Normalization [7]. Decoder layers incorporate an additional cross-attention mechanism placed between the multi-head self-attention and the position-wise FFNs. Additionally, modifications are made to the self-attention mechanisms in the decoder to prevent each position from attending future positions. A visual representation of the Transformer architecture can be found in Fig. A.2.

The Transformer employs an attention mechanism utilizing the Query-Key-

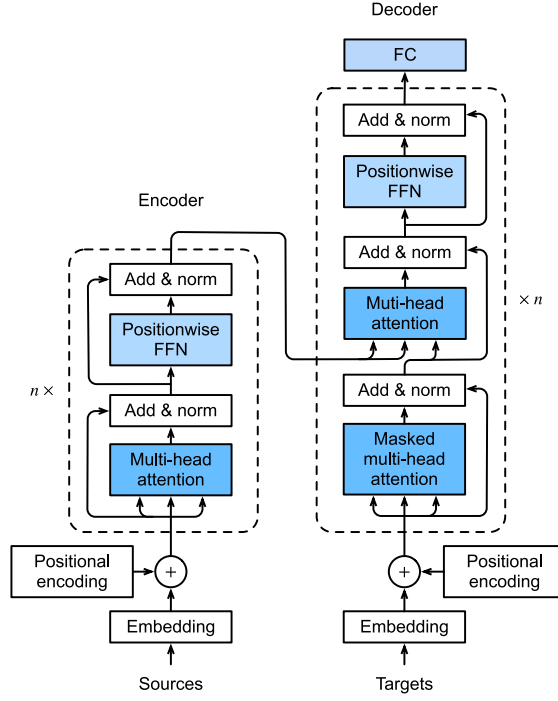


Figure A.2: The Transformer architecture [314]. Source: [363].

Value (QKV) paradigm. Given matrix representations for queries $Q \in \mathbb{R}^{N \times D_k}$, keys $K \in \mathbb{R}^{M \times D_k}$, and values $V \in \mathbb{R}^{M \times D_v}$, the Transformer scaled dot-product attention is expressed as:

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{D_k}} \right) V = AV, \quad (\text{A.2})$$

where N and M represent the lengths of the queries and keys (or values), respectively; D_k and D_v indicate the dimensions of the keys (or queries) and the values. The matrix $A = \text{softmax} \left(\frac{QK^T}{\sqrt{D_k}} \right)$ is often referred to as the attention matrix. The softmax function is applied row-wise. To alleviate the gradient vanishing issue associated with the softmax function, the dot-products of the queries and keys are scaled by the factor $\sqrt{D_k}$.

Instead of relying on a singular attention function, the Transformer utilizes a multi-head attention mechanism. In this approach, the original queries, keys, and values of dimension D_m are projected into dimensions D_k , D_k and D_v using H distinct learned projections. The attention output is computed for each set of these projected queries, keys, and values based on the attention of Eq. A.2. All outputs are then concatenated and re-projected to form a D_m -dimensional

A.2. ARCHITECTURES

representation:

$$\begin{aligned} \text{MultiHeadAttn}(Q, K, V) &= \text{Concat}(\text{head}_1, \dots, \text{head}_H)W^O, \\ \text{where head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V). \end{aligned} \quad (\text{A.3})$$

Within the Transformer architecture, attention is categorized based on the source of queries and key-value pairs, resulting in three distinct types:

- **Self-attention:** In the encoder of the Transformer, the queries, keys, and values are all set to X , which represents the output from the preceding layer, as described in Eq. A.3.
- **Masked Self-attention:** The decoder self-attention in the Transformer is designed such that queries at a given position can only focus on key-value pairs up to that specific position. To allow for parallelized training, an attention mask is typically applied to the unnormalized attention matrix $\hat{A} = \exp\left(\frac{QK^T}{\sqrt{D_k}}\right)$. Here, any illegal positions are masked by setting $\hat{A}_{ij} = -\infty$ if $i < j$. This form of self-attention is frequently termed as autoregressive or causal attention.
- **Cross-attention:** Here, the queries are derived from the outputs of the preceding decoder layer, while the keys and values are obtained using the encoder outputs.

The position-wise feed-forward network (FFN) operates individually on each position. It is mathematically represented as:

$$\text{FFN}(H') = \text{ReLU}(H'W^1 + b^1)W^2 + b^2 \quad (\text{A.4})$$

where H' denotes the output from the preceding layer, and $W^1 \in \mathbb{R}^{D_m \times D_f}$, $W^2 \in \mathbb{R}^{D_f \times D_m}$, $b^1 \in \mathbb{R}^{D_f}$, and $b^2 \in \mathbb{R}^{D_m}$ are the parameters to be learned. Notably, the intermediary dimension D_f of the feed-forward network is often set larger than D_m .

As already mentioned, to construct a deep architecture, the Transformer utilizes a residual connection around each component, succeeded by Layer Normalization. Each Transformer encoder block can then be formulated as:

$$H' = \text{LayerNorm}(\text{SelfAttention}(X) + X) \quad (\text{A.5})$$

$$H = \text{LayerNorm}(\text{FFN}(H') + H') \quad (\text{A.6})$$

where $\text{SelfAttention}()$ denotes the self-attention mechanism and $\text{LayerNorm}()$ represents the layer normalization operation.

The Transformer, in the absence of recurrence or convolution, lacks inherent positional information, particularly for the encoder. This necessitates the incorporation of an additional positional representation to capture the ordering of tokens. This positional information is represented using absolute sinusoidal position encodings. For each position index t , the encoding is represented as a vector $p_t = \text{PE}(t) \in \mathbb{R}^{D_m}$, where each element is either a sinusoidal or cosine function of the index, determined by a predefined frequency:

$$\text{PE}(t)_i = \begin{cases} \sin(\omega_i t) & \text{if } i \text{ is even} \\ \cos(\omega_i t) & \text{if } i \text{ is odd} \end{cases} \quad (\text{A.7})$$

where ω_i denotes the hand-crafted frequency for each dimension. The position encoding for each position in the sequence is then added to the token embeddings prior to being fed to the Transformer.

Generally, the Transformer architecture can be utilized in three distinct ways:

- **Encoder–Decoder.** The complete Transformer architecture is employed. This is commonly used for sequence-to-sequence modeling tasks, such as neural machine translation.
- **Encoder only.** Only the encoder is utilized, and the outputs of the encoder serve as a representation for the input sequence. This approach is frequently adopted for natural language understanding tasks, including text classification and sequence labeling, and in vision tasks.
- **Decoder only.** Only the decoder component is used, with the encoder-decoder cross-attention module being removed. This approach is mainly used for sequence generation tasks, such as language modeling.

Transformers are frequently compared with CNNs. Convolutional networks inherently possess the inductive biases of translation equivariance and locality due to their shared local kernel functions. In contrast, the Transformer architecture makes no assumptions regarding the structural information of data. For this reason, the Transformer is a universal and adaptable architecture. How-

A.2. ARCHITECTURES

ever, a consequential drawback is that the absence of structural bias makes the Transformer prone to overfitting, especially with limited data.

References

- [1] Evan Ackerman. *Self-driving cars were just around the corner - in 1960*. Accessed: October 2023. 2016. URL: <https://spectrum.ieee.org/tech-history/heroic-failures/selfdriving-cars-were-just-around-the-cornerin-1960>.
- [2] Ateret Anaby-Tavor et al. "Do not have enough data? Deep learning to the rescue!" In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 05. 2020, pp. 7383–7390.
- [3] Moab Arar et al. "Unsupervised Multi-Modal Image Registration via Geometry Preserving Image-to-Image Translation". In: *CVPR*. 2020.
- [4] Martín Arjovsky and Léon Bottou. "Towards Principled Methods for Training Generative Adversarial Networks". In: *ICLR*. OpenReview.net, 2017.
- [5] Martín Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein GAN". In: *ICML* (2017).
- [6] Anurag Arnab and Philip H. S. Torr. "Pixelwise Instance Segmentation with a Dynamically Instantiated Network". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.
- [7] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. "Layer Normalization". In: *CoRR* abs/1607.06450 (2016).
- [8] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation". In: *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)* 39.12 (2017), pp. 2481–2495.
- [9] Kyungjune Baek et al. "Rethinking the Truly Unsupervised Image-to-Image Translation". In: *arXiv* (2020).

REFERENCES

- [10] David Bau et al. “Seeing What a GAN Cannot Generate”. In: *ICCV*. IEEE, 2019, pp. 4501–4510.
- [11] Sven Bauer, Yasamin Alkhorshid, and Gerd Wanielik. “Using high-definition maps for precise urban vehicle localization”. In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. 2016, pp. 492–497.
- [12] R. Behringer. “The darpa grand challenge - autonomous ground vehicles in the desert”. In: *IFAC/EURON Symposium on Intelligent Autonomous Vehicles*. Vol. 37. 8. Lisbon, Portugal, July 2004, pp. 904–909.
- [13] Iz Beltagy, Matthew E. Peters, and Arman Cohan. “Longformer: The Long-Document Transformer”. In: *CoRR abs/2004.05150* (2020).
- [14] Sagie Benaïm and Lior Wolf. “One-Sided Unsupervised Domain Mapping”. In: *NIPS*. 2017, pp. 752–762.
- [15] Michele Bertoncello and Dominik Wee. *Ten ways autonomous driving could redefine the automotive world*. Tech. rep. 2015.
- [16] Massimo Bertozzi et al. “Stereo vision-based approaches for pedestrian detection”. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)-Workshops*. 2005, pp. 16–16.
- [17] Massimo Bertozzi et al. “Stereo vision-based vehicle detection”. In: *Proceedings of the IEEE Intelligent Vehicles Symposium 2000*. 2000, pp. 39–44.
- [18] Aditya Bhat. “Autonomous vehicles: A perspective of past and future trends”. In: *International Journal of Engineering Technology Science and Research* 4.10 (2017).
- [19] Deblina Bhattacharjee et al. “DUNIT: Detection-Based Unsupervised Image-to-Image Translation”. In: *CVPR*. 2020.
- [20] Mikolaj Binkowski et al. “Demystifying MMD GANs”. In: *ICLR (Poster)*. 2018.
- [21] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. “Variational inference: A review for statisticians”. In: *Journal of the American statistical Association* (2017).
- [22] P. Blyth et al. “Expanding the design horizon for self-driving vehicles: Distributing benefits and burdens”. In: *IEEE Technology and Society Magazine* 35.3 (Sept. 2016), pp. 44–49.

- [23] Ali Borji. “Pros and cons of GAN evaluation measures”. In: *Comput. Vis. Image Underst.* 179 (2019), pp. 41–65.
- [24] Andrew Brock, Jeff Donahue, and Karen Simonyan. “Large Scale GAN Training for High Fidelity Natural Image Synthesis”. In: *ICLR*. 2019.
- [25] A. Broggi, M. Bertozzi, and A. Fascioli. “Argo and the millemiglia in automatico tour”. In: *IEEE Intelligent Systems and their Applications* 14.1 (1999), pp. 55–64.
- [26] A. Broggi et al. “Shape-based Pedestrian Detection”. In: *Proc. IEEE Intelligent Vehicles Symposium (IV)*. 2000.
- [27] Alberto Broggi et al. “Autonomous vehicles control in the vislab inter-continental autonomous challenge”. In: *Annual Reviews in Control* 36.1 (2012), pp. 161–171.
- [28] Michael M. Bronstein et al. “Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges”. In: *CoRR* abs/2104.13478 (2021).
- [29] Tom B. Brown et al. “Language Models are Few-Shot Learners”. In: *NeurIPS*. 2020.
- [30] Tom Bruls et al. “Generating all the roads to rome: Road layout randomization for improved road marking segmentation”. In: *ITSC*. 2019.
- [31] Jie Cao et al. “ReMix: Towards Image-to-Image Translation with Limited Data”. In: *CVPR*. 2021.
- [32] Weihan Cao et al. “Pkd: General distillation framework for object detectors via pearson correlation coefficient”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 15394–15406.
- [33] Zhangjie Cao, Qixing Huang, and Karthik Ramani. “3D Object Classification via Spherical Projections”. In: *3DV*. IEEE Computer Society, 2017, pp. 566–574.
- [34] Nicolas Carion et al. “End-to-End Object Detection with Transformers”. In: *ECCV (1)*. Vol. 12346. Lecture Notes in Computer Science. Springer, 2020, pp. 213–229.
- [35] Daniele Cattaneo, Matteo Vaghi, and Abhinav Valada. “Lcdnet: Deep loop closure detection and point cloud registration for lidar slam”. In: *IEEE Transactions on Robotics* (2022).

REFERENCES

- [36] Rohan Chabra et al. “Deep Local Shapes: Learning Local SDF Priors for Detailed 3D Reconstruction”. In: *ECCV*. Springer, 2020.
- [37] Kelvin C. K. Chan et al. “GLEAN: Generative Latent Bank for Image Super-Resolution and Beyond”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 45.3 (2023), pp. 3154–3168.
- [38] Defang Chen et al. “Cross-Layer Distillation with Semantic Calibration”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2021, pp. 7028–7036.
- [39] Defang Chen et al. “Knowledge Distillation with the Reused Teacher Classifier”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 11933–11942.
- [40] Jingwen Chen et al. “Image Blind Denoising With Generative Adversarial Network Based Noise Modeling”. In: *CVPR*. Computer Vision Foundation / IEEE Computer Society, 2018, pp. 3155–3164.
- [41] Kai Chen et al. “MMDetection: Open MMLab Detection Toolbox and Benchmark”. In: *arXiv preprint arXiv:1906.07155* (2019).
- [42] Liang-Chieh Chen et al. “DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs”. In: *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)* 40.4 (2018), pp. 834–848.
- [43] Liang-Chieh Chen et al. “Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation”. In: *ECCV*. 2018.
- [44] Liang-Chieh Chen et al. “MaskLab: Instance Segmentation by Refining Object Detection With Semantic and Direction Features”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 4013–4022.
- [45] Liang-Chieh Chen et al. “Rethinking Atrous Convolution for Semantic Image Segmentation”. In: *arXiv.org abs/1706.05587* (2017).
- [46] Liang-Chieh Chen et al. “Rethinking atrous convolution for semantic image segmentation”. In: *arXiv preprint arXiv:1706.05587* (2017).
- [47] Lili Chen et al. “Decision Transformer: Reinforcement Learning via Sequence Modeling”. In: *NeurIPS*. 2021, pp. 15084–15097.

- [48] Xiaozhi Chen et al. “Monocular 3d object detection for autonomous driving”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2147–2156.
- [49] Xiaozhi Chen et al. “Multi-View 3D Object Detection Network for Autonomous Driving”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.
- [50] An-Chieh Cheng et al. “Learning 3D Dense Correspondence via Canonical Point Autoencoder”. In: *Advances in Neural Information Processing Systems* 34 (2021).
- [51] Anoop Cherian and Alan Sullivan. “Sem-GAN: Semantically-consistent image-to-image translation”. In: *WACV*. 2019.
- [52] Hsu-kuang Chiu et al. “Probabilistic 3D multi-modal, multi-object tracking for autonomous driving”. In: *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2021, pp. 14227–14233.
- [53] Yunjey Choi et al. “Stargan: Unified generative adversarial networks for multi-domain image-to-image translation”. In: *CVPR*. 2018.
- [54] Laurene Claussmann et al. “A review of motion planning for highway autonomous driving”. In: *IEEE Transactions on Intelligent Transportation Systems* 21.5 (2019), pp. 1826–1848.
- [55] Adam Coates, Andrew Ng, and Honglak Lee. “An analysis of single-layer networks in unsupervised feature learning”. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings*. 2011, pp. 215–223.
- [56] Felipe Codevilla et al. “End-to-end driving via conditional imitation learning”. In: *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2018, pp. 4693–4700.
- [57] MMSegmentation Contributors. *MMSegmentation: OpenMMLab Semantic Segmentation Toolbox and Benchmark*. <https://github.com/open-mmlab/mms Segmentation>. 2020.
- [58] Aurélien Cord and Didier Aubert. “Towards rain detection through use of in-vehicle multipurpose cameras”. In: *IV*. 2011.

REFERENCES

- [59] Marius Cordts et al. “The cityscapes dataset for semantic urban scene understanding”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3213–3223.
- [60] Marius Cordts et al. “The cityscapes dataset for semantic urban scene understanding”. In: *CVPR*. 2016.
- [61] Arthur Daniel Costea, Andra Petrovai, and Sergiu Nedevschi. “Fusion Scheme for Semantic and Instance-level Segmentation”. In: *Proc. IEEE Conf. on Intelligent Transportation Systems (ITSC)*. 2018, pp. 3469–3475.
- [62] Florinel-Alin Croitoru et al. “Diffusion Models in Vision: A Survey”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 45.9 (2023), pp. 10850–10869.
- [63] Jifeng Dai, Kaiming He, and Jian Sun. “Instance-Aware Semantic Segmentation via Multi-Task Network Cascades”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2016.
- [64] Anthony Dell’Eva, Marco Orsingher, and Massimo Bertozzi. “Arbitrary Point Cloud Upsampling with Spherical Mixture of Gaussians”. In: *3DV. IEEE*, 2022, pp. 465–474.
- [65] Anthony Dell’Eva et al. “Leveraging Local Domains for Image-to-Image Translation”. In: *VISIGRAPP (5: VISAPP)*. SCITEPRESS, 2022, pp. 179–189.
- [66] Anthony Dell’Eva et al. “Teacher Features-Driven Regularization for Knowledge Distillation”. In: *Under review*. 2023.
- [67] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.
- [68] Emily L. Denton et al. “Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks”. In: *NIPS*. 2015, pp. 1486–1494.
- [69] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *NAACL-HLT (1)*. Association for Computational Linguistics, 2019, pp. 4171–4186.
- [70] Prafulla Dhariwal and Alexander Quinn Nichol. “Diffusion Models Beat GANs on Image Synthesis”. In: *NeurIPS*. 2021, pp. 8780–8794.

- [71] Juergen Dickmann et al. "Automotive radar the key technology for autonomous driving: From detection and ranging to environmental understanding". In: *2016 IEEE Radar Conference (RadarConf)*. 2016, pp. 1–6.
- [72] Ernst Dickmanns et al. "The seeing passenger car 'vamos-p'". In: *Proceedings of the Intelligent Vehicles '94 Symposium*. 1994, pp. 68–73.
- [73] Xiaohan Ding et al. "Centripetal sgD for pruning very deep convolutional networks with complicated structure". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4943–4953.
- [74] Xin Dong, Shangyu Chen, and Sinno Pan. "Learning to prune deep neural networks via layer-wise optimal brain surgeon". In: *Advances in neural information processing systems* 30 (2017).
- [75] Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *ICLR*. 2021.
- [76] Xinxin Du et al. "A General Pipeline for 3D Detection of Vehicles". In: *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*. 2018, pp. 3194–3200.
- [77] Vijay Prakash Dwivedi and Xavier Bresson. "A Generalization of Transformer Networks to Graphs". In: *CoRR abs/2012.09699* (2020).
- [78] Andreas Eitel et al. "Multimodal deep learning for robust RGB-D object recognition". In: *IROS. IEEE*, 2015, pp. 681–687.
- [79] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. "Neural architecture search: A survey". In: *The Journal of Machine Learning Research* 20.1 (2019), pp. 1997–2017.
- [80] Nico Engel, Vasileios Belagiannis, and Klaus Dietmayer. "Point transformer". In: *IEEE Access* 9 (2021), pp. 134826–134840.
- [81] Carl Engelking. *The 'driverless' car era began more than 90 years ago*. Accessed: October 2023. 2017. URL: <https://www.discovermagazine.com/technology/the-driverless-car-era-began-more-than-90-years-ago>.
- [82] D. Fagnant and K. Knockelman. *Preparing a nation for autonomous vehicles*. Tech. rep. Eno Center for Transportation, 2013.

REFERENCES

- [83] Gongfan Fang et al. “Depgraph: Towards any structural pruning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 16091–16101.
- [84] Pedro F. Felzenszwalb, David A. McAllester, and Deva Ramanan. “A discriminatively trained, multiscale, deformable part model”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2008.
- [85] Di Feng, Lars Rosenbaum, and Klaus Dietmayer. “Towards safe autonomous driving: Capture uncertainty in the deep neural network for lidar 3d vehicle detection”. In: *2018 21st international conference on intelligent transportation systems (ITSC)*. 2018, pp. 3266–3273.
- [86] Wanquan Feng et al. “Neural Points: Point Cloud Representation with Neural Fields”. In: *arXiv preprint arXiv:2112.04148* (2021).
- [87] Raghudeep Gadde et al. “Efficient 2D and 3D Facade Segmentation Using Auto-Context”. In: *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)* 40.5 (2018), pp. 1273–1280.
- [88] Tomer Galanti, Lior Wolf, and Sagie Benaim. “The Role of Minimal Complexity Functions in Unsupervised Learning of Semantic Mappings”. In: *ICLR (Poster)*. OpenReview.net, 2018.
- [89] Mudasir A Ganaie et al. “Ensemble deep learning: A review”. In: *Engineering Applications of Artificial Intelligence* 115 (2022), p. 105151.
- [90] Ganzorig Gankhuyag et al. “Lightweight Real-Time Image Super-Resolution Network for 4K Images”. In: *CVPR Workshops*. IEEE, 2023, pp. 1746–1755.
- [91] Andreas Geiger et al. “Vision meets Robotics: The KITTI Dataset”. In: *International Journal of Robotics Research (IJRR)* (2013).
- [92] Golnaz Ghiasi and Charless C. Fowlkes. “Laplacian Pyramid Reconstruction and Refinement for Semantic Segmentation”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2016.
- [93] Amir Gholami et al. “A survey of quantization methods for efficient neural network inference”. In: *arXiv preprint arXiv:2103.13630* (2021).
- [94] Ross Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2014.

- [95] Ross B. Girshick. “Fast R-CNN”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2015.
- [96] S. Alireza Golestaneh and Lina J. Karam. “Spatially-Varying Blur Detection Based on Multiscale Fused and Sorted Transform Coefficients of Gradient Magnitudes”. In: *CVPR*. 2017.
- [97] Rui Gong et al. “Dlow: Domain flow for adaptation and generalization”. In: *CVPR*. 2019.
- [98] Xinyu Gong et al. “AutoGAN: Neural Architecture Search for Generative Adversarial Networks”. In: *ICCV*. IEEE, 2019, pp. 3223–3233.
- [99] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. Vol. 1. Cambridge: MIT Press, 2016.
- [100] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: <http://www.deeplearningbook.org>.
- [101] Ian J. Goodfellow et al. “Generative Adversarial Networks”. In: *NeurIPS* (2014).
- [102] Jianping Gou et al. “Knowledge distillation: A survey”. In: *International Journal of Computer Vision* 129 (2021), pp. 1789–1819.
- [103] Thibault Groueix et al. “A papier-mâché approach to learning 3d surface generation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 216–224.
- [104] Anmol Gulati et al. “Conformer: Convolution-augmented Transformer for Speech Recognition”. In: *INTERSPEECH*. ISCA, 2020, pp. 5036–5040.
- [105] Ishaan Gulrajani et al. “Improved Training of Wasserstein GANs”. In: *NIPS*. 2017, pp. 5767–5777.
- [106] Chuan Guo et al. “On calibration of modern neural networks”. In: *International conference on machine learning*. PMLR. 2017, pp. 1321–1330.
- [107] Meng-Hao Guo et al. “Pct: Point cloud transformer”. In: *Computational Visual Media* 7.2 (2021), pp. 187–199.
- [108] Yiwen Guo, Anbang Yao, and Yurong Chen. “Dynamic network surgery for efficient dnns”. In: *Advances in neural information processing systems* 29 (2016).

REFERENCES

- [109] Ziyao Guo et al. “Class Attention Transfer Based Knowledge Distillation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 11868–11877.
- [110] Timo Hackel, Jan D. Wegner, and Konrad Schindler. “Fast semantic segmentation of 3d point clouds with strongly varying density”. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences (APRS)* III-3 (2016), pp. 177–184.
- [111] Shirsendu Sukanta Halder, Jean-François Lalonde, and Raoul de Charette. “Physics-Based Rendering for Improving Robustness to Rain”. In: *ICCV*. 2019.
- [112] Zeeshan Hayder, Xuming He, and Mathieu Salzmann. “Boundary-Aware Instance Segmentation”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 587–595.
- [113] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.
- [114] Kaiming He et al. “Mask r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.
- [115] Kaiming He et al. “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2014.
- [116] Byeongho Heo et al. “A comprehensive overhaul of feature distillation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 1921–1930.
- [117] Martin Heusel et al. “Gans trained by a two time-scale update rule converge to a local nash equilibrium”. In: *NeurIPS*. 2017.
- [118] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. “Distilling the Knowledge in a Neural Network”. In: *NIPS Deep Learning and Representation Learning Workshop*. 2015.
- [119] Gao Huang et al. “Densely Connected Convolutional Networks”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2261–2269.

- [120] Huaibo Huang et al. “IntroVAE: Introspective Variational Autoencoders for Photographic Image Synthesis”. In: *NeurIPS*. 2018.
- [121] Hui Huang et al. “Consolidation of unorganized point clouds for surface reconstruction”. In: *ACM transactions on graphics (TOG)* 28.5 (2009), pp. 1–7.
- [122] Hui Huang et al. “Edge-aware point set resampling”. In: *ACM transactions on graphics (TOG)* 32.1 (2013), pp. 1–12.
- [123] Keli Huang et al. “Multi-modal Sensor Fusion for Auto Driving Perception: A Survey”. In: *CoRR* abs/2202.02703 (2022).
- [124] Tao Huang et al. “Masked Distillation with Receptive Tokens”. In: *The Eleventh International Conference on Learning Representations*. 2023. URL: <https://openreview.net/forum?id=mWRngkvIki3>.
- [125] Xun Huang et al. “Multimodal unsupervised image-to-image translation”. In: *ECCV*. 2018.
- [126] D. H. Hubel and T. N. Wiesel. “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex”. In: *J Physiol* 160.1 (1962), p. 106.
- [127] Audi initiative. *The Pulse of Autonomous Driving (white paper)*. Tech. rep. Audi AG, 2019.
- [128] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *ICML*. Vol. 37. JMLR Workshop and Conference Proceedings. JMLR.org, 2015, pp. 448–456.
- [129] Phillip Isola et al. “Image-to-image translation with conditional adversarial networks”. In: *CVPR*. 2017.
- [130] Joel Janai et al. “Computer Vision for Autonomous Vehicles: Problems, Datasets and State of the Art”. In: *Found. Trends Comput. Graph. Vis.* 12.1-3 (2020), pp. 1–308.
- [131] Maximilian Jaritz et al. “End-to-end race driving with deep reinforcement learning”. In: *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2018, pp. 2070–2075.

REFERENCES

- [132] Simon Jégou et al. “The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2017, pp. 1175–1183.
- [133] Hyeok-June Jeong, Kyeong-Sik Park, and Young-Guk Ha. “Image preprocessing for efficient training of YOLO deep learning networks”. In: *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*. IEEE. 2018, pp. 635–637.
- [134] Ying Jin, Jiaqi Wang, and Dahua Lin. “Multi-Level Logit Distillation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 24276–24285.
- [135] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. “Perceptual Losses for Real-Time Style Transfer and Super-Resolution”. In: *ECCV (2)*. Vol. 9906. Lecture Notes in Computer Science. Springer, 2016, pp. 694–711.
- [136] Alexia Jolicoeur-Martineau. “On Relativistic f-Divergences”. In: *ICML*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 4931–4939.
- [137] Alexia Jolicoeur-Martineau. “The relativistic discriminator: a key element missing from standard GAN”. In: *ICLR (Poster)*. 2019.
- [138] Chanyong Jung, Gihyun Kwon, and Jong Chul Ye. “Exploring Patch-wise Semantic Relation for Contrastive Learning in Image-to-Image Translation Tasks”. In: *CVPR*. IEEE, 2022, pp. 18239–18248.
- [139] Divas Karimanzira et al. “Object detection in sonar images”. In: *Electronics* 9.7 (2020), p. 1180.
- [140] Tero Karras, Samuli Laine, and Timo Aila. “A Style-Based Generator Architecture for Generative Adversarial Networks”. In: *CVPR*. Computer Vision Foundation / IEEE, 2019, pp. 4401–4410.
- [141] Tero Karras et al. “Alias-Free Generative Adversarial Networks”. In: *NeurIPS*. 2021, pp. 852–863.
- [142] Tero Karras et al. “Analyzing and Improving the Image Quality of StyleGAN”. In: *CVPR*. Computer Vision Foundation / IEEE, 2020, pp. 8107–8116.

- [143] Tero Karras et al. “Progressive Growing of GANs for Improved Quality, Stability, and Variation”. In: *ICLR*. OpenReview.net, 2018.
- [144] Bernhard Kerbl et al. “3D Gaussian Splatting for Real-Time Radiance Field Rendering”. In: *ACM Trans. Graph.* 42.4 (2023), 139:1–139:14.
- [145] Taeksoo Kim et al. “Learning to Discover Cross-Domain Relations with Generative Adversarial Networks”. In: *ICML*. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 1857–1865.
- [146] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *ICLR (Poster)*. 2015.
- [147] Thomas N. Kipf and Max Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. In: *ICLR (Poster)*. 2017.
- [148] Alexander Kirillov et al. “InstanceCut: From Edges to Instances with MultiCut”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 7322–7331.
- [149] Alexander Kirillov et al. “Panoptic Feature Pyramid Networks”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [150] Alexander Kirillov et al. “Panoptic Segmentation”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 9404–9413.
- [151] Jelena Kocić, Nenad Jovičić, and Vujo Drndarević. “Sensors and sensor fusion in autonomous vehicles”. In: *2018 26th Telecommunications Forum (TELFOR)*. 2018, pp. 420–425.
- [152] Jason Kong et al. “Kinematic and dynamic vehicle models for autonomous driving control design”. In: *2015 IEEE intelligent vehicles symposium (IV)*. IEEE. 2015, pp. 1094–1099.
- [153] Alex Krizhevsky, Geoffrey Hinton, et al. “Learning multiple layers of features from tiny images”. In: (2009).
- [154] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.
- [155] Jason Ku, Alex D. Pon, and Steven L. Waslander. “Monocular 3D Object Detection Leveraging Accurate Proposals and Shape Reconstruction”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019.

REFERENCES

- [156] Taylor Kubota. *Stanford's robotics legacy*. Accessed: October 2023. 2019. URL: <https://news.stanford.edu/2019/01/16/stanfords-robotics-legacy/>.
- [157] Abhijit Kundu, Yin Li, and James M. Rehg. "3D-RCNN: Instance-Level 3D Object Reconstruction via Render-and-Compare". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 3559–3568.
- [158] Sampo Kuutti et al. "A survey of the state-of-the-art localization techniques and their potentials for autonomous vehicle applications". In: *IEEE Internet of Things Journal* 5.2 (2018), pp. 829–846.
- [159] Tuomas Kynkäänniemi et al. "Improved Precision and Recall Metric for Assessing Generative Models". In: *NeurIPS*. 2019, pp. 3929–3938.
- [160] Guillaume Lample et al. "Fader Networks: Manipulating Images by Sliding Attributes". In: *NeurIPS*. 2017.
- [161] Francesco Laneve, Alessandro Rucco, and Massimo Bertozzi. "A real-time collision-free maneuver generation algorithm for autonomous driving". In: *European Journal of Control* (2023).
- [162] Francesco Laneve, Alessandro Rucco, and Massimo Bertozzi. "A Trajectory Optimization Strategy for Merging Maneuvers of Autonomous Vehicles". In: *CONTROLO 2022*. Springer International Publishing, 2022, pp. 3–14.
- [163] Hei Law and Jia Deng. "CornerNet: Detecting Objects as Paired Keypoints". In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2018, pp. 765–781.
- [164] Ya Le and Xuan Yang. "Tiny imagenet visual recognition challenge". In: *CS 231N 7.7* (2015), p. 3.
- [165] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.
- [166] Cheng-Han Lee et al. "MaskGAN: Towards Diverse and Interactive Facial Image Manipulation". In: *CVPR*. Computer Vision Foundation / IEEE, 2020, pp. 5548–5557.
- [167] Hanbit Lee, Jinseok Seol, and Sang-goo Lee. "Contrastive Learning for Unsupervised Image-to-Image Translation". In: *arXiv* (2021).

- [168] Hsin-Ying Lee et al. "Drit++: Diverse image-to-image translation via disentangled representations". In: *IJCV* (2020).
- [169] Namhoon Lee et al. "A Signal Propagation Perspective for Pruning Neural Networks at Initialization". In: *ICLR*. 2020.
- [170] Jesse Levinson and Sebastian Thrun. "Robust vehicle localization in urban environments using probabilistic maps". In: *2010 IEEE international conference on robotics and automation*. 2010, pp. 4372–4378.
- [171] Bo Li, Tianlei Zhang, and Tian Xia. "Vehicle Detection from 3D Lidar Using Fully Convolutional Network". In: *Proc. Robotics: Science and Systems (RSS)*. 2016.
- [172] Hao Li et al. "Pruning Filters for Efficient ConvNets". In: *ICLR (Poster)*. 2017.
- [173] Peiliang Li, Xiaozhi Chen, and Shaojie Shen. "Stereo r-cnn based 3d object detection for autonomous driving". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 7644–7652.
- [174] Peilun Li et al. "Semantic-aware grad-gan for virtual-to-real urban scene adaption". In: *BMVC* (2018).
- [175] Ruihui Li et al. "Point cloud upsampling via disentangled refinement". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 344–353.
- [176] Ruihui Li et al. "Pu-gan: a point cloud upsampling adversarial network". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 7203–7212.
- [177] Ruihui Li et al. "SP-GAN: Sphere-Guided 3D Shape Generation and Manipulation". In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 40.4 (2021).
- [178] Yi Li et al. "Fully Convolutional Instance-Aware Semantic Segmentation". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 4438–4446.
- [179] Ying Li et al. "Deep learning for lidar point clouds in autonomous driving: A review". In: *IEEE Transactions on Neural Networks and Learning Systems* 32.8 (2020), pp. 3412–3432.

REFERENCES

- [180] You Li and Javier Ibanez-Guzman. “Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems”. In: *IEEE Signal Processing Magazine* 37.4 (2020), pp. 50–61.
- [181] Jae Hyun Lim and Jong Chul Ye. “Geometric GAN”. In: *CoRR* abs/1705.02894 (2017).
- [182] Che-Tsung Lin et al. “Multimodal Structure-Consistent Image-to-Image Translation.” In: *AAAI*. 2020.
- [183] Jianxin Lin et al. “ZstGAN: An Adversarial Approach for Unsupervised Zero-Shot Image-to-Image Translation”. In: *Neurocomputing* (2021).
- [184] Tsung-Yi Lin et al. “Focal loss for dense object detection”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.
- [185] Tsung-Yi Lin et al. “Microsoft COCO: Common Objects in Context”. In: *ECCV*. Vol. 8693. Lecture Notes in Computer Science. Springer, 2014, pp. 740–755.
- [186] Yaron Lipman et al. “Parameterization-free projection for geometry reconstruction”. In: *ACM Transactions on Graphics (TOG)* 26.3 (2007), 22–es.
- [187] Liyang Liu et al. “Group fisher pruning for practical network compression”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 7021–7032.
- [188] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. “Unsupervised image-to-image translation networks”. In: *NeurIPS*. 2017.
- [189] Ming-Yu Liu and Oncel Tuzel. “Coupled Generative Adversarial Networks”. In: *NIPS*. 2016, pp. 469–477.
- [190] Ming-Yu Liu et al. “Few-shot unsupervised image-to-image translation”. In: *ICCV*. 2019.
- [191] Wei Liu et al. “SSD: Single Shot MultiBox Detector”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2016.
- [192] Yahui Liu et al. “Describe what to change: A text-guided unsupervised image-to-image translation approach”. In: *ACM MM*. 2020.
- [193] Yifan Liu et al. “Structured knowledge distillation for semantic segmentation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 2604–2613.

- [194] Yinhan Liu et al. “RoBERTa: A Robustly Optimized BERT Pretraining Approach”. In: *CoRR* abs/1907.11692 (2019).
- [195] Ze Liu et al. “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows”. In: *ICCV*. IEEE, 2021, pp. 9992–10002.
- [196] Zhuang Liu et al. “A convnet for the 2020s”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 11976–11986.
- [197] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully Convolutional Networks for Semantic Segmentation”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2015.
- [198] Ilya Loshchilov and Frank Hutter. “Decoupled Weight Decay Regularization”. In: *International Conference on Learning Representations*. 2019.
- [199] Ilya Loshchilov and Frank Hutter. “SGDR: Stochastic Gradient Descent with Warm Restarts”. In: *International Conference on Learning Representations*. 2017.
- [200] Chenxu Luo, Xiaodong Yang, and Alan Yuille. “Exploring simple 3d multi-object tracking for autonomous driving”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10488–10497.
- [201] Chenxu Luo, Xiaodong Yang, and Alan Yuille. “Self-supervised pillar motion learning for autonomous driving”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 3183–3192.
- [202] Luqing Luo et al. “PU-EVA: An Edge-Vector Based Approximation Solution for Flexible-Scale Point Cloud Upsampling”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 16208–16217.
- [203] Wenhan Luo et al. “Multiple object tracking: A literature review”. In: *Artificial intelligence* 293 (2021), p. 103448.
- [204] Liqian Ma et al. “Exemplar guided unsupervised image-to-image translation with semantic consistency”. In: *ICLR*. 2019.
- [205] Saturnino Maldonado-Bascón et al. “Road-sign detection and recognition based on support vector machines”. In: *IEEE transactions on intelligent transportation systems* 8.2 (2007), pp. 264–278.

REFERENCES

- [206] Xudong Mao et al. “Least Squares Generative Adversarial Networks”. In: *ICCV*. 2017.
- [207] Andelo Martinović et al. “3D All The Way: Semantic Segmentation of Urban Scenes from Start to End in 3D”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2015.
- [208] Kirill Mazur and Victor Lempitsky. “Cloud transformers: A universal approach to point cloud processing tasks”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10715–10724.
- [209] Warren S McCulloch and Walter Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.
- [210] Youssef Alami Mejjati et al. “Unsupervised attention-guided image-to-image translation”. In: *NeurIPS*. 2018.
- [211] Lars M. Mescheder et al. “Occupancy Networks: Learning 3D Reconstruction in Function Space”. In: *CVPR*. Computer Vision Foundation / IEEE, 2019, pp. 4460–4470.
- [212] Ben Mildenhall et al. “NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis”. In: *ECCV*. Springer, 2020.
- [213] Yue Ming et al. “Deep learning for monocular depth estimation: A review”. In: *Neurocomputing* 438 (2021), pp. 14–33.
- [214] Seyed Iman Mirzadeh et al. “Improved knowledge distillation via teacher assistant”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 04. 2020, pp. 5191–5198.
- [215] Ishan Misra, Rohit Girdhar, and Armand Joulin. “An End-to-End Transformer Model for 3D Object Detection”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 2906–2917.
- [216] Takeru Miyato et al. “Spectral Normalization for Generative Adversarial Networks”. In: *ICLR*. 2018.
- [217] Sangwoo Mo, Minsu Cho, and Jinwoo Shin. “Instagan: Instance-aware image-to-image translation”. In: *ICLR* (2019).
- [218] W David Montgomery. *Public and private benefits of autonomous vehicles*. Tech. rep. 2018.

- [219] W David Montgomery et al. *America's workforce and the self-driving future: Realizing productivity gains and spurring economic growth*. Tech. rep. 2018.
- [220] Reza Moradi, Reza Berangi, and Behrouz Minaei. "A survey of regularization strategies for deep models". In: *Artificial Intelligence Review* 53 (2020), pp. 3947–3986.
- [221] Thomas Müller et al. "Instant neural graphics primitives with a multiresolution hash encoding". In: *ACM Trans. Graph.* 41.4 (2022), 102:1–102:15.
- [222] Muhammad Ferjad Naeem et al. "Reliable Fidelity and Diversity Metrics for Generative Models". In: *ICML*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 7176–7185.
- [223] Thu Nguyen-Phuoc et al. "Hologan: Unsupervised learning of 3d representations from natural images". In: *ICCV*. 2019.
- [224] Alexander Quinn Nichol et al. "GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models". In: *ICML*. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 16784–16804.
- [225] Maria-Elena Nilsback and Andrew Zisserman. "Automated Flower Classification over a Large Number of Classes". In: *Indian Conference on Computer Vision, Graphics and Image Processing*. 2008.
- [226] Yuval Nirkin, Yosi Keller, and Tal Hassner. "FSGAN: Subject Agnostic Face Swapping and Reenactment". In: *ICCV*. IEEE, 2019, pp. 7183–7192.
- [227] Augustus Odena, Christopher Olah, and Jonathon Shlens. "Conditional Image Synthesis with Auxiliary Classifier GANs". In: *ICML*. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 2642–2651.
- [228] Michael Oechsle, Songyou Peng, and Andreas Geiger. "UNISURF: Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View Reconstruction". In: *ICCV*. IEEE, 2021, pp. 5569–5579.
- [229] Marco Orsingher et al. "Revisiting PatchMatch Multi-View Stereo for Urban 3D Reconstruction". In: *IV*. IEEE, 2022, pp. 190–196.
- [230] Xingang Pan et al. "Spatial As Deep: Spatial CNN for Traffic Scene Understanding". In: *AAAI*. 2017.

REFERENCES

- [231] Guan Pang and Ulrich Neumann. “3D point cloud object detection with multi-view convolutional neural network”. In: *ICPR*. IEEE, 2016, pp. 585–590.
- [232] Jiahao Pang, Duanshun Li, and Dong Tian. “TearingNet: Point Cloud Autoencoder to Learn Topology-Friendly Representations”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.
- [233] Jeong Joon Park et al. “DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation”. In: *CVPR*. Computer Vision Foundation / IEEE, 2019, pp. 165–174.
- [234] Sejun Park et al. “Lookahead: A Far-sighted Alternative of Magnitude-based Pruning”. In: *ICLR*. 2020.
- [235] Taesung Park et al. “Semantic image synthesis with spatially-adaptive normalization”. In: *CVPR*. 2019.
- [236] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.
- [237] Or Patashnik et al. “BalaGAN: Cross-Modal Image Translation Between Imbalanced Domains”. In: *CVPR Workshops*. 2021.
- [238] Songyou Peng et al. “Convolutional Occupancy Networks”. In: *ECCV (3)*. Vol. 12348. Lecture Notes in Computer Science. Springer, 2020, pp. 523–540.
- [239] Chen Pengguang et al. “Distilling Knowledge via Knowledge Review”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.
- [240] Praveena Penmetsa et al. “Perceptions and expectations of autonomous vehicles – A snapshot of vulnerable road user opinion”. In: *Technological Forecasting and Social Change* 143 (2019), pp. 9–13. ISSN: 0040-1625. DOI: <https://doi.org/10.1016/j.techfore.2019.02.010>. URL: <http://www.sciencedirect.com/science/article/pii/S0040162518316603>.
- [241] Bojan Pepik et al. “Multi-View and 3D Deformable Part Models”. In: *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)* 37.11 (2015), pp. 2232–2245.

- [242] Anna Petrovskaya and Sebastian Thrun. “Model based vehicle tracking for autonomous driving in urban environments”. In: *Proceedings of robotics: science and systems IV, Zurich, Switzerland* 34 (2008).
- [243] Tobias Pfaff et al. “Learning Mesh-Based Simulation with Graph Networks”. In: *ICLR*. 2021.
- [244] Pedro H. O. Pinheiro, Ronan Collobert, and Piotr Dollár. “Learning to Segment Object Candidates”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2015, pp. 1990–1998.
- [245] Pedro Oliveira Pinheiro et al. “Learning to Refine Object Segments”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2016, pp. 75–91.
- [246] Fabio Pizzati, Pietro Cerri, and Raoul de Charette. “CoMoGAN: continuous model-guided image-to-image translation”. In: *CVPR*. 2021.
- [247] Fabio Pizzati, Pietro Cerri, and Raoul de Charette. “Physics-Informed Guided Disentanglement in Generative Networks”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 45.8 (2023), pp. 10300–10316.
- [248] Tobias Pohlen et al. “Full-Resolution Residual Networks for Semantic Segmentation in Street Scenes”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 3309–3318.
- [249] Dean Pomerleau and Todd Jochem. *No hands across america journal*. Accessed: October 23. 1995. URL: <https://www.cs.cmu.edu/~tjochem/nhaa/Journal.html>.
- [250] Charles R Qi et al. “Pointnet: Deep learning on point sets for 3d classification and segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 652–660.
- [251] Charles Ruizhongtai Qi et al. “Pointnet++: Deep hierarchical feature learning on point sets in a metric space”. In: *Advances in neural information processing systems* 30 (2017).
- [252] Guocheng Qian et al. “Pu-gcn: Point cloud upsampling using graph convolutional networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 11683–11692.
- [253] Yue Qian et al. “Deep Magnification-Flexible Upsampling Over 3D Point Clouds”. In: *IEEE Transactions on Image Processing* 30 (2021), pp. 8354–8367.

REFERENCES

- [254] Shi Qiu, Saeed Anwar, and Nick Barnes. “PU-Transformer: Point Cloud Upsampling Transformer”. In: *arXiv preprint arXiv:2111.12242* (2021).
- [255] A. Radford et al. “Language models are unsupervised multitask learners”. In: *OpenAI Blog* (2019).
- [256] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”. In: *ICLR*. 2016.
- [257] Aditya Ramesh et al. “Hierarchical Text-Conditional Image Generation with CLIP Latents”. In: *CoRR* abs/2204.06125 (2022).
- [258] Pierluigi Zama Ramirez, Alessio Tonioni, and Luigi Di Stefano. “Exploiting semantics in adversarial training for image-level domain adaptation”. In: *IPAS*. 2018.
- [259] Joseph Redmon and Ali Farhadi. “YOLO9000: Better, Faster, Stronger”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.
- [260] Joseph Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788.
- [261] Shaoqing Ren et al. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *Advances in neural information processing systems* 28 (2015).
- [262] Ricardo Ribani and Mauricio Marengoni. “A survey of transfer learning for convolutional neural networks”. In: *2019 32nd SIBGRAPI conference on graphics, patterns and images tutorials (SIBGRAPI-T)*. IEEE. 2019, pp. 47–57.
- [263] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. “OctNet: Learning Deep 3D Representations at High Resolutions”. In: *CVPR*. IEEE Computer Society, 2017, pp. 6620–6629.
- [264] Eduardo Romera et al. “Bridging the day and night domain gap for semantic segmentation”. In: *IV*. 2019.
- [265] Adriana Romero et al. “FitNets: Hints for Thin Deep Nets”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: <http://arxiv.org/abs/1412.6550>.

- [266] Frank Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain”. In: *Psychological review* 65.6 (1958), p. 386.
- [267] Kevin Roth et al. “Stabilizing Training of Generative Adversarial Networks through Regularization”. In: *NIPS*. 2017, pp. 2018–2028.
- [268] Sebastian Ruder. “An overview of gradient descent optimization algorithms”. In: *CoRR* abs/1609.04747 (2016).
- [269] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. “Learning representations by back-propagating errors”. In: *nature* 323.6088 (1986), pp. 533–536.
- [270] Kuniaki Saito, Kate Saenko, and Ming-Yu Liu. “COCO-FUNIT: Few-Shot Unsupervised Image Translation with a Content Conditioned Style Encoder”. In: *ECCV*. 2020.
- [271] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. “ACDC: The Adverse Conditions Dataset with Correspondences for Semantic Driving Scene Understanding”. In: *ICCV*. 2021.
- [272] Tim Salimans et al. “Improved techniques for training gans”. In: *NeurIPS*. 2016.
- [273] Victor Sanh, Thomas Wolf, and Alexander Rush. “Movement pruning: Adaptive sparsity by fine-tuning”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 20378–20389.
- [274] S. Scagliarini. *Smart roads e driverless cars: tra diritto, tecnologie, etica pubblica*. G. Giappichelli Editore, 2019.
- [275] Mark Schutera et al. “Night-to-Day: Online Image-to-Image Translation for Object Detection Within Autonomous Driving by Night”. In: *IEEE T-IV* (2020).
- [276] Brent Schwarz. “Mapping the world in 3D”. In: *Nature Photonics* 4.7 (2010), pp. 429–430.
- [277] Ramprasaath R Selvaraju et al. “Grad-cam: Visual explanations from deep networks via gradient-based localization”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 618–626.

REFERENCES

- [278] Sunando Sengupta et al. "Urban 3D Semantic Modelling Using Stereo Vision". In: *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*. 2013.
- [279] Pierre Sermanet et al. "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks". In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2014.
- [280] Pierre Sermanet et al. "Pedestrian Detection with Unsupervised Multi-stage Feature Learning". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2013.
- [281] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. "Singan: Learning a generative model from a single natural image". In: *ICCV*. 2019.
- [282] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.
- [283] Zhiqiang Shen and Eric Xing. "A fast knowledge distillation framework for visual recognition". In: *European Conference on Computer Vision*. Springer. 2022, pp. 673–690.
- [284] Zhiqiang Shen et al. "Towards instance-level image-to-image translation". In: *CVPR*. 2019.
- [285] Ziyi Shen et al. "Deep Semantic Face Deblurring". In: *CVPR. Computer Vision Foundation / IEEE Computer Society*, 2018, pp. 8260–8269.
- [286] Connor Shorten and Taghi M Khoshgoftaar. "A survey on image data augmentation for deep learning". In: *Journal of big data* 6.1 (2019), pp. 1–48.
- [287] Ajay Shrestha and Ausif Mahmood. "Review of deep learning algorithms and architectures". In: *IEEE access* 7 (2019), pp. 53040–53065.
- [288] Changyong Shu et al. "Channel-wise knowledge distillation for dense prediction". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 5311–5320.
- [289] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *ICLR*. 2015.
- [290] Ayan Sinha, Jing Bai, and Karthik Ramani. "Deep Learning 3D Shape Surfaces Using Geometry Images". In: *ECCV (6)*. Vol. 9910. Lecture Notes in Computer Science. Springer, 2016, pp. 223–240.

- [291] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. “Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations”. In: *NeurIPS*. 2019, pp. 1119–1130.
- [292] Akash Srivastava et al. “VEEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning”. In: *NIPS*. 2017, pp. 3308–3318.
- [293] Hang Su et al. “Multi-view Convolutional Neural Networks for 3D Shape Recognition”. In: *ICCV*. IEEE Computer Society, 2015, pp. 945–953.
- [294] Shunqiao Sun, Athina P Petropulu, and H Vincent Poor. “MIMO radar for advanced driver-assistance systems and autonomous driving: Advantages and challenges”. In: *IEEE Signal Processing Magazine* 37.4 (2020), pp. 98–117.
- [295] Christian Szegedy et al. “Going deeper with convolutions”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2015.
- [296] Domen Tabernik and Danijel Skočaj. “Deep learning for large-scale traffic-sign detection and recognition”. In: *IEEE transactions on intelligent transportation systems* 21.4 (2019), pp. 1427–1440.
- [297] Talha Takleh Omar Takleh et al. “A brief survey on SLAM methods in autonomous vehicle”. In: *International Journal of Engineering and Technology* 7.4 (2018), pp. 38–43.
- [298] Ardi Tampuu et al. “A survey of end-to-end driving: Architectures and training methods”. In: *IEEE Transactions on Neural Networks and Learning Systems* 33.4 (2020), pp. 1364–1384.
- [299] Mingxing Tan and Quoc Le. “Efficientnet: Rethinking model scaling for convolutional neural networks”. In: *International conference on machine learning*. PMLR. 2019, pp. 6105–6114.
- [300] Matthew Tancik et al. “Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains”. In: *NeurIPS* (2020).
- [301] Hao Tang et al. “Multi-Channel Attention Selection GANs for Guided Image-to-Image Translation”. In: *CVPR*. 2019.
- [302] Siyu Teng et al. “Motion planning for autonomous driving: The state of the art and future perspectives”. In: *IEEE Transactions on Intelligent Vehicles* (2023).

REFERENCES

- [303] Sebastian Thrun et al. “Stanley: The robot that won the darpa grand challenge”. In: *Journal of field Robotics* 23.9 (2006), pp. 661–692.
- [304] Yonglong Tian, Dilip Krishnan, and Phillip Isola. “Contrastive Representation Distillation”. In: *ICLR*. 2020.
- [305] Maxime Tremblay et al. “Rain rendering for evaluating and improving robustness to bad weather”. In: *IJCV* (2020).
- [306] Frederick Tung and Greg Mori. “Similarity-preserving knowledge distillation”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 1365–1374.
- [307] *TuSimple Benchmark*. <https://github.com/TuSimple/tusimple-benchmark>. Accessed: August 2021.
- [308] Jasper RR Uijlings et al. “Selective search for object recognition”. In: *International Journal of Computer Vision (IJCV)* 104.2 (2013), pp. 154–171.
- [309] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. “Instance Normalization: The Missing Ingredient for Fast Stylization”. In: *CoRR* abs/1607.08022 (2016).
- [310] C. Urmson and W. Whittaker. “Self-driving cars and the urban challenge”. In: *IEEE Intelligent Systems* 23.2 (Mar. 2008), pp. 66–68.
- [311] Chris Urmson et al. “Tartan racing: A multi-modal approach to the darpa urban challenge”. In: (2007).
- [312] Mikaela Angelina Uy et al. “Revisiting Point Cloud Classification: A New Benchmark Dataset and Classification Model on Real-World Data”. In: *International Conference on Computer Vision (ICCV)*. 2019.
- [313] Girish Varma et al. “IDD: A Dataset for Exploring Problems of Autonomous Navigation in Unconstrained Environments”. In: *WACV*. 2019.
- [314] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).
- [315] Petar Velickovic et al. “Graph Attention Networks”. In: *ICLR (Poster)*. OpenReview.net, 2018.
- [316] *Visionair*. <http://www.infra-visionair.eu/>. Accessed: May 2022.
- [317] Dan Wang et al. “Multi-view 3d reconstruction with transformers”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 5722–5731.

- [318] Dominic Zeng Wang and Ingmar Posner. “Voting for Voting in Online Point Cloud Object Detection”. In: *Proc. Robotics: Science and Systems (RSS)*. 2015.
- [319] Nanyang Wang et al. “Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images”. In: *ECCV (11)*. Vol. 11215. Springer, 2018, pp. 55–71.
- [320] Peng Wang et al. “NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction”. In: *NeurIPS*. 2021, pp. 27171–27183.
- [321] Ting-Chun Wang et al. “Few-shot Video-to-Video Synthesis”. In: *NeurIPS*. 2019, pp. 5014–5025.
- [322] Ting-Chun Wang et al. “High-Resolution Image Synthesis and Semantic Manipulation With Conditional GANs”. In: *CVPR. Computer Vision Foundation / IEEE Computer Society*, 2018, pp. 8798–8807.
- [323] Xinlong Wang et al. “SOLO: Segmenting Objects by Locations”. In: *Proc. Eur. Conf. Computer Vision (ECCV)*. 2020.
- [324] Xintao Wang et al. “Deep network interpolation for continuous imagery effect transition”. In: *CVPR*. 2019.
- [325] Yaxing Wang et al. “Semi-supervised learning for few-shot image-to-image translation”. In: *CVPR*. 2020.
- [326] Yue Wang et al. “Dynamic Graph CNN for Learning on Point Clouds”. In: *ACM Transactions on Graphics (TOG)* (2019).
- [327] Yue Wang et al. “Pillar-based object detection for autonomous driving”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII* 16. Springer. 2020, pp. 18–34.
- [328] Zhangjing Wang, Yu Wu, and Qingqing Niu. “Multi-Sensor Fusion in Automated Driving: A Survey”. In: *IEEE Access* 8 (2020), pp. 2847–2868.
- [329] Steven Euijong Whang and Jae-Gil Lee. “Data collection and quality challenges for deep learning”. In: *Proceedings of the VLDB Endowment* 13.12 (2020), pp. 3429–3432.
- [330] Bichen Wu et al. “SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud”. In: *ICRA. IEEE*, 2018, pp. 1887–1893.

REFERENCES

- [331] Wayne Wu et al. “Transgaga: Geometry-aware unsupervised image-to-image translation”. In: *CVPR*. 2019.
- [332] Yuxin Wu and Kaiming He. “Group Normalization”. In: *ECCV (13)*. Vol. 11217. Lecture Notes in Computer Science. Springer, 2018, pp. 3–19.
- [333] Zhirong Wu et al. “3D ShapeNets: A deep representation for volumetric shapes”. In: *CVPR*. IEEE Computer Society, 2015, pp. 1912–1920.
- [334] Zifeng Wu, Chunhua Shen, and Anton van den Hengel. “Wider or Deeper: Revisiting the ResNet Model for Visual Recognition”. In: *Pattern Recognition* 90 (2019), pp. 119–133.
- [335] Wei Xiao et al. “Rule-based optimal control for autonomous driving”. In: *Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems*. 2021, pp. 143–154.
- [336] Enze Xie et al. “SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers”. In: *NeurIPS*. 2021, pp. 12077–12090.
- [337] Mingle Xu et al. “Instance-Level Image Translation with a Local Discriminator”. In: *IEEE Access* (2021).
- [338] Kento Yabuuchi et al. “Visual Localization for Autonomous Driving using Pre-built Point Cloud Maps”. In: *2021 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2021, pp. 913–919.
- [339] Jiancheng Yang et al. “Modeling point clouds with self-attention and gumbel subset sampling”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3323–3332.
- [340] Jing Yang et al. “Knowledge distillation via softmax regression representation learning”. In: *International Conference on Learning Representations*. 2020.
- [341] Shuai Yang et al. “GP-UNIT: Generative Prior for Versatile Unsupervised Image-to-Image Translation”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 45.10 (2023), pp. 11869–11883.
- [342] Yaoqing Yang et al. “Foldingnet: Point cloud auto-encoder via deep grid deformation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 206–215.

- [343] Ze Yang et al. “RepPoints: Point Set Representation for Object Detection”. In: *The IEEE International Conference on Computer Vision (ICCV)*. Oct. 2019.
- [344] Zhendong Yang et al. “Focal and global knowledge distillation for detectors”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 4643–4652.
- [345] Zhendong Yang et al. “Masked generative distillation”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 53–69.
- [346] Lior Yariv et al. “Volume Rendering of Neural Implicit Surfaces”. In: *NeurIPS*. 2021, pp. 4805–4815.
- [347] Shuquan Ye et al. “Meta-PU: An arbitrary-scale upsampling network for point cloud”. In: *IEEE Transactions on Visualization and Computer Graphics* (2021).
- [348] De Jong Yeong et al. “Sensor and Sensor Fusion Technology in Autonomous Vehicles: A Review”. In: *Sensors* 21.6 (2021), p. 2140.
- [349] De Jong Yeong et al. “Sensor and sensor fusion technology in autonomous vehicles: A review”. In: *Sensors* 21.6 (2021), p. 2140.
- [350] Wang Yifan et al. “Patch-based progressive 3d point set upsampling”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5958–5967.
- [351] Yurong You et al. “Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving”. In: *arXiv preprint arXiv:1906.06310* (2019).
- [352] Zhonghui You et al. “Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks”. In: *Advances in neural information processing systems* 32 (2019).
- [353] Fisher Yu and Vladlen Koltun. “Multi-Scale Context Aggregation by Dilated Convolutions”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2016.
- [354] Fisher Yu et al. *Berkeley DeepDrive*. <https://bdd-data.berkeley.edu/>. Accessed: October 2023. 2019.
- [355] Jiahui Yu et al. “Free-Form Image Inpainting with Gated Convolution”. In: *ICCV*. 2019.

REFERENCES

- [356] Lequan Yu et al. “Ec-net: an edge-aware point set consolidation network”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 386–402.
- [357] Lequan Yu et al. “Pu-net: Point cloud upsampling network”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 2790–2799.
- [358] Xumin Yu et al. “PointR: Diverse point cloud completion with geometry-aware transformers”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 12498–12507.
- [359] Zehao Yu et al. “MonoSDF: Exploring Monocular Geometric Cues for Neural Implicit Surface Reconstruction”. In: *NeurIPS*. 2022.
- [360] Yuhui Yuan, Xilin Chen, and Jingdong Wang. “Object-Contextual Representations for Semantic Segmentation”. In: *ECCV*. 2020.
- [361] Sergey Zagoruyko and Nikos Komodakis. “Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer”. In: *ICLR*. 2017. URL: <https://arxiv.org/abs/1612.03928>.
- [362] Sergey Zagoruyko and Nikos Komodakis. “Wide Residual Networks”. In: *Proceedings of the British Machine Vision Conference 2016, BMVC*. BMVA Press, 2016.
- [363] Aston Zhang et al. *Dive into Deep Learning*. <https://D2L.ai>. Cambridge University Press, 2023.
- [364] Richard Zhang et al. “The unreasonable effectiveness of deep features as a perceptual metric”. In: *CVPR*. 2018.
- [365] Yihuan Zhang et al. “Road-segmentation-based curb detection method for self-driving via a 3D-lidar sensor”. In: *IEEE transactions on intelligent transportation systems* 19.12 (2018), pp. 3981–3991.
- [366] Ying Zhang et al. “Deep mutual learning”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4320–4328.
- [367] Ziyu Zhang, Sanja Fidler, and Raquel Urtasun. “Instance-Level Segmentation for Autonomous Driving with Deep Densely Connected MRFs”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2016.

- [368] Ziyu Zhang et al. “Monocular Object Instance Segmentation and Depth Ordering with CNNs”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2015.
- [369] Borui Zhao et al. “Decoupled knowledge distillation”. In: *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*. 2022, pp. 11953–11962.
- [370] Hengshuang Zhao et al. “Point transformer”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 16259–16268.
- [371] Hengshuang Zhao et al. “PSANet: Point-wise Spatial Attention Network for Scene Parsing”. In: *ECCV*. 2018.
- [372] Hengshuang Zhao et al. “Pyramid Scene Parsing Network”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6230–6239.
- [373] Hengshuang Zhao et al. “Pyramid scene parsing network”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2881–2890.
- [374] Shengyu Zhao et al. “Large Scale Image Completion via Co-Modulated Generative Adversarial Networks”. In: *ICLR*. 2021.
- [375] Yiqin Zhao and Tian Guo. “Pointar: Efficient lighting estimation for mobile augmented reality”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 678–693.
- [376] Haitian Zheng et al. “CM-GAN: Image Inpainting with Cascaded Modulation GAN and Object-Aware Training”. In: *ECCV*. 2022.
- [377] Tu Zheng et al. “RESA: Recurrent Feature-Shift Aggregator for Lane Detection”. In: *AAAI*. 2021.
- [378] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. “Open3D: A Modern Library for 3D Data Processing”. In: *arXiv:1801.09847* (2018).
- [379] Sharon Zhou et al. “HYPE: A Benchmark for Human eYe Perceptual Evaluation of Generative Models”. In: *NeurIPS*. 2019, pp. 3444–3456.
- [380] Yin Zhou and Oncel Tuzel. “VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

REFERENCES

- [381] Jun-Yan Zhu et al. "Toward multimodal image-to-image translation". In: *NeurIPS*. 2017.
- [382] Jun-Yan Zhu et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks". In: *CVPR*. 2017.
- [383] Peihao Zhu et al. "SEAN: Image Synthesis with Semantic Region-Adaptive Normalization". In: *CVPR*. 2020.
- [384] Zhen Zhu et al. "Semantically Multi-modal Image Synthesis". In: *CVPR*. 2020.
- [385] M.Z. Zia et al. "Detailed 3D Representations for Object Recognition and Modeling". In: *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)* 35.11 (Nov. 2013), pp. 2608–2623.
- [386] M.Zeeshan Zia, Michael Stark, and Konrad Schindler. "Towards Scene Understanding with Detailed 3D Object Representations". In: *International Journal of Computer Vision (IJCV)* 112.2 (2015), pp. 188–203.
- [387] Saman Zia et al. "RGB-D Object Recognition Using Deep Convolutional Neural Networks". In: *ICCV Workshops*. IEEE Computer Society, 2017, pp. 887–894.