DISS. ETH NO. 30227

2D AND 3D GENERATIVE MODELS UNDER REAL-WORLD CONSTRAINTS

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES
(Dr. sc. ETH Zurich)

presented by

MOHAMAD SHAHBAZI
M.Sc., Sharif University of Technology

born on 4 February 1994

accepted on the recommendation of

Prof. Dr. Luc Van Gool, examiner
Prof. Dr. Jun-Yan Zhu, co-examiner
Dr. Timo Aila, co-examiner
Dr. Anna Khoreva, co-examiner
Dr. Danda Pani Paudel, co-examiner

2024

To my Family and my Friends

ABSTRACT

Recent advancements in generative modeling have transformed visual content creation, showing tremendous promise in several applications in Computer Vision and Graphics. However, the adoption of generative models in everyday tasks is hindered by challenges in controllability of the generation process, data requirements, and computational demands. This thesis focuses on addressing such real-world constraints in 2D and 3D generative models.

Firstly, we focus on improving the data efficiency of class-conditional Generative Adversarial Networks (GANs) using transfer learning. We introduce a new class-specific transfer learning method, called cGANTransfer, to explicitly propagate the knowledge from old classes to the new ones based on their relevance. Through extensive evaluation, we demonstrate the superiority of the proposed approach over the previous methods for conditional GAN transfer.

Secondly, we investigate the training of class-conditional GANs with small datasets. In particular, we identify conditioning collapse in GANs–mode collapse caused by conditional GAN training on small data. We propose a training strategy based on transitional conditioning that effectively prevents the observed mode collapse by additionally leveraging unconditional learning. The proposed method results not only in stable training but also in generating high-quality images, thanks to the exploitation of shared information across classes in the early stages of training.

Thirdly, we tackle the computational efficiency of NeRF-GANs, a class of 3D-aware generative models based on the integration of Neural Radiance Fields (NeRFs) and GANs, trained on single-view image datasets. Specifically, we revisit pose-conditioned 2D GANs for efficient 3D-aware generation at inference time by distilling 3D knowledge from pretrained NeRF-GANs. We propose a simple and effective method for efficient inference of 3D-aware GANs, based on re-using the well-disentangled latent space of a pre-trained NeRF-GAN in a pose-conditioned convolutional network, to directly generate 3D-consistent images corresponding to the underlying 3D representations.

Lastly, we address the novel task of object generation in 3D scenes without the need for any 3D supervision or 3D placement guidance from the users. We introduce InseRF, a novel method for generative object insertion in the NeRF reconstructions of 3D scenes. Based on a user-provided textual description and only a 2D bounding box in a reference viewpoint, InseRF is capable of controllable and 3D-consistent object insertion in 3D scenes without requiring explicit 3D information as input.

ZUSAMMENFASSUNG

Jüngste Fortschritte in der generativen Modellierung haben die Erstellung visueller Inhalte revolutioniert und zeigen erhebliches Potenzial in verschiedenen Anwendungen der Computer Vision und Grafik. Die Nutzung generativer Modelle für alltägliche Aufgaben wird jedoch durch Herausforderungen in der Steuerbarkeit des Generierungsprozesses, den Datenanforderungen und dem Rechenaufwand erschwert. Diese Dissertation konzentriert sich darauf, solche realen Einschränkungen in 2D- und 3D-generativen Modellen zu adressieren.

Erstens konzentrieren wir uns auf die Verbesserung der Dateneffizienz von class-conditional Generative Adversarial Networks (GANs) durch Transfer Learning. Wir stellen eine neue klassenspezifische Transfer-Learning-Methode namens cGAN-Transfer vor, die das Wissen explizit von alten zu neuen Klassen auf Basis ihrer Relevanz überträgt. Umfangreiche Auswertungen zeigen, dass der vorgeschlagene Ansatz früheren Methoden für den bedingten GAN-Transfer überlegen ist.

Zweitens untersuchen wir das Training von class-conditional GANs mit kleinen Datensätzen. Insbesondere identifizieren wir den Bedingungskollaps (conditioning collapse) in GANs—einen Modus-Kollaps, der durch das Training bedingter GANs mit kleinen Datenmengen verursacht wird. Wir schlagen eine Trainingsstrategie vor, die auf Übergangsbedingungen basiert und den beobachteten Modus-Kollaps durch zusätzliches unbedingtes Lernen (unconditional learning) effektiv verhindert. Die vorgeschlagene Methode führt nicht nur zu stabilem Training, sondern auch zur Erzeugung hochqualitativer Bilder, indem in den frühen Trainingsphasen geteilte Informationen über Klassen hinweg genutzt werden.

Drittens befassen wir uns mit der Recheneffizienz von NeRF-GANs, einer Klasse von 3D-bewussten generativen Modellen, die auf der Integration von Neural Radiance Fields (NeRFs) und GANs basieren und auf Single-View-Datensätzen trainiert werden. Insbesondere überarbeiten wir pose-conditionierte 2D-GANs für eine effiziente 3D-bewusste Bilderzeugung zur Laufzeit, indem wir 3D-Wissen aus vortrainierten NeRF-GANs destillieren. Wir schlagen eine einfache und effektive Methode für eine effiziente Inferenz 3D-bewusster GANs vor, die auf der Wiederverwendung des gut disentangled latenten Raums eines vortrainierten NeRF-GANs in einem pose-conditionierten konvolutionalen Netzwerk basiert, um direkt 3D-konsistente Bilder entsprechend den zugrundeliegenden 3D-Repräsentationen zu erzeugen.

Abschließend behandeln wir die neuartige Aufgabe der Objekterzeugung in 3D-Szenen ohne die Notwendigkeit einer 3D-Supervision oder 3D-Platzierungsanweisungen

von Benutzern. Wir stellen InseRF vor, eine neuartige Methode zur generativen Einfügung von Objekten in die NeRF-Rekonstruktionen von 3D-Szenen. Basierend auf einer vom Benutzer bereitgestellten Textbeschreibung und nur einem 2D-Begrenzungsrahmen in einer Referenzansicht ist InseRF in der Lage, Objekte kontrolliert und 3D-konsistent in 3D-Szenen einzufügen, ohne explizite 3D-Informationen als Eingabe zu benötigen.

PUBLICATIONS

The following publications are included in this thesis (* denotes shared authorship):

- Mohamad Shahbazi, Zhiwu Huang, Danda Pani Paudel, Ajad Chhatkuli, and Luc Van Gool. "Efficient Conditional GAN Transfer with Knowledge Propagation across Classes." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- Mohamad Shahbazi, Martin Danelljan, Danda Pani Paudel, and Luc Van Gool. "Collapse by Conditioning: Training Class-conditional GANs with Limited Data." In *International Conference on Learning Representations (ICLR)*, 2022.
- Mohamad Shahbazi, Evangelos Ntavelis, Alessio Tonioni, Edo Collins, Danda Pani Paudel, Martin Danelljan, and Luc Van Gool. "NeRF-GAN Distillation for Efficient 3D-Aware Generation with Convolutions." In *IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, 2023.
- Mohamad Shahbazi*, Liesbeth Claessens*, Michael Niemeyer, Edo Collins, Alessio Tonioni, Luc Van Gool, and Federico Tombari. "InseRF: Text-Driven Generative Object Insertion in Neural 3D Scenes." *Under review*, 2024.

The following publications are also parts of the Ph.D. research, but they are not included in this thesis:

- Evangelos Ntavelis, Mohamad Shahbazi, Iason Kastanis, Radu Timofte, Martin Danelljan, and Luc Van Gool. "Arbitrary-Scale Image Synthesis." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- Zilin Fang, Mohamad Shahbazi, Thomas Probst, Danda Pani Paudel, and Luc Van Gool. "Training Dynamics Aware Neural Network Optimization with Stabilization." In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, 2022.
- Chuqiao Li, Zhiwu Huang, Danda Pani Paudel, Yabin Wang, Mohamad Shahbazi, Xiaopeng Hong, and Luc Van Gool. "A Continual Deepfake Detection Benchmark: Dataset, Methods, and Essentials." In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2023.

- Shengqu Cai, Eric Ryan Chan, Songyou Peng, Mohamad Shahbazi, Anton Obukhov, Luc Van Gool, and Gordon Wetzstein. "DiffDreamer: Towards Consistent Unsupervised Single-view Scene Extrapolation with Conditional Diffusion Models." In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

- Evangelos Ntavelis, Mohamad Shahbazi, Iason Kastanis, Radu Timofte, Martin Danelljan, and Luc Van Gool. "StyleGenes: Discrete and Efficient Latent Distributions for GANs." In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2024.

- Saeed Khorram, Mingqi Jiang, Mohamad Shahbazi, Mohamad H. Danesh, and Li Fuxin. "Taming the Tail in Class-Conditional GANs: Knowledge Sharing via Unconditional Training at Lower Resolutions." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

ACKNOWLEDGEMENTS

First, I extend my heartfelt gratitude to Prof. Luc Van Gool for the opportunity to pursue a PhD at CVL. His exceptional support and guidance have been pivotal throughout my PhD journey, and working under his supervision has been truly an honor for me. I am also grateful to my co-advisors Danda Pani Paudel and Martin Danelljan, who have been fantastic mentors and friends to me. I have learned so much from them and have grown significantly as a result of their mentorship.

I would also like to thank my collaborators at CVL. It was a great pleasure to work with Evan Ntavelis, whose energy and personality made research much more delightful. I also thank Ajad Chhatkuli, Zhiwu Huang, and Thomas Probst for their insights and help.

Next, I would like to thank all my friends who made my time at CVL exceptionally enjoyable, especially Goutam, Prune, Sara, Nikola, Edo, Christoph, Eric, Niko, David, Menelaus, and Anton, among many others. I am also thankful to Dr. Kristine Haberer, Christine Braun-Roth, and Christina Krueger for their indispensable support and for making the PhD journey smoother for us at CVL.

I am grateful to Federico Tombari for providing me with the opportunity to work with his team at Google Zurich as an intern. I also thank my collaborators at Google, Alessio Tonioni, Michael Niemeyer, Edo Collins, and Liesbeth Claessens, who made my experience pleasant and joyful. I am also thankful to the amazing team at Microsoft Reading, especially Eric Sommerlade, Alexandros Neophytou, and Marcelo Gennari do Nascimento, whom I had the pleasure of working with during my internship.

I extend my appreciation to the members of my PhD examination committee, Prof. Jun-Yan Zhu, Dr. Timo Aila, and Dr. Anna Khoreva, for the time and the effort they invested in the evaluation of my PhD thesis.

Above all, my deepest gratitude is reserved for my family. I am forever grateful to my loving parents, Maryam and Roohollah, who have been the bedrock of my progress and growth. I am immensely thankful to my dear sister, Mahya, and my brother-in-law, Farokh, for their unwavering support and inspiration. I am truly fortunate to have such a loving and supportive family, and my love and appreciation for them is beyond what words can express.

CONTENTS

1

INTRODUCTION

1.1 INTRODUCTION TO GENERATIVE MODELS

Generative models are a class of machine learning models designed to learn the underlying probability distribution of input data. This allows them to generate novel samples that resemble the original data. Generative modeling in computer vision and graphics has seen rapid progress in recent years. The growing interest in generative models stems from their potential to transform digital art and content creation for applications such as animation, gaming, and augmented/virtual reality. Digital visual content creation typically requires hours of painstaking manual work by artists and graphics experts, who often rely on tools based on handcrafted and imperfect models of the visual world. Generative models, however, have shown great promise in significantly decreasing the need for human interaction, and they are able to discover the underlying structure of the visual content implicitly from real-world data, for more realistic content generation.

The promise of generative models extends beyond content creation. Generative models hold tremendous potential for representation learning without the need for extensive data annotations. For instance, generative models can be used for dataset synthesis and augmentation to increase the diversity of training data, in setups where collecting labelled data is challenging [1–3]. Moreover, generative models have the ability to extract rich, hierarchical representations of data with no explicit supervision. Such representations have recently been shown to be significantly beneficial in unsupervised and semi-supervised visual recognition tasks. The representations obtained from generative models have either been directly exploited for unsupervised visual understanding [4, 5], or they have been used as strong initial priors along with additional supervision with real or even synthetic data [6–8].

The history of generative modeling traces its roots back to pre-deep learning techniques, where models such as Hidden Markov Models (HMMs) and Gaussian Mixture Models (GMMs) played a pivotal role in understanding and generating data [9, 10]. Moreover, Bayesian Networks [11] were integral in the early landscape of generative models, providing a foundation for probabilistic reasoning in data analysis. Building upon these foundational concepts, generative modeling has seen a significant evolution with the rise of deep learning. There are various deep learning-based paradigms for generative modeling, which can be mainly categorized as autoregressive models [12, 13], variational auto-encoders (VAEs) [14], generative

adversarial networks (GANs) [15], flow-based models [16], and the more recent denoising diffusion models (DDMs) [17]. Among them, GANs and DDMs have particularly achieved impressive results in a variety of applications involving the synthesis and editing of realistic images, videos, and 3D scenes.

GANs, introduced in the seminal work by Goodfellow et al. [15], revolutionized visual generative modeling by achieving unprecedented realism of the generated content. For several years, GANs dominated the area of generative modeling, enabling a variety of applications, including unconditional and class-conditioned generation [18, 19], image-to-image translation [20, 21], style transfer [22], semantic editing [23], and text-to-image generation [24]. However, the adversarial dynamics of GANs make them prone to training instability and lack of diversity (mode collapse), especially when trained on small datasets [25].

Recently, denoising diffusion models [17] have brought about a new wave of breakthroughs in generative modeling, overtaking GANs in many generative tasks. Especially, DDMs have achieved extraordinary capabilities in the task of text-to-image generation [26], unlocking a plethora of applications in computer vision and graphics. In addition to generative applications, DDMs are making their way into several image understanding tasks thanks to their rich and strong priors [7, 8, 27]. When it comes to computational efficiency, however, DDMs are currently limited in their generation speed due to their iterative denoising process. Many efforts have already been made to improve the efficiency of DDMs [26, 28–30]. Notably, the integration of DDMs and GANs through adversarial distillation has been shown promising in multiple studies [30, 31], combining the stable training of diffusion models with the efficient inference of GAN-based generators.

1.2 GENERATIVE MODELS UNDER REAL-WORLD CONSTRAINTS

A significant amount of effort has been dedicated to increasing the quality and complexity of the content generated by generative models [32, 33]. However, there remains a gap to be filled when it comes to the practical use of such models by everyday users in their tasks:

1.2.1 *Controllability*

To start with, controllability is an important prerequisite to the real-world utilization of generative models. Controllability in generative models pertains to the degree of influence or guidance users can exert over the output, ensuring that the generated content aligns with specific criteria or preferences. Such control can be provided in various forms of model conditioning, such as target categories [18], textual de-

scriptions [26], images [20], semantic layouts [21], and geometric information [34]. Users should be able to interact with generative models with control signals that are convenient and intuitive, while providing enough flexibility to steer the generation.

1.2.2 *Adaptation to custom data*

Another crucial element in making generative models accessible to a broader community is their ability to adapt to custom data. Generative models often require training on large-scale datasets to achieve desirable performance [25]. In many real-world applications, however, gathering large amounts of data is challenging and costly. Data collection becomes even more difficult in the case of controllable generation, where the samples are required to be paired with additional annotations. In scene understanding tasks, there exist numerous established techniques for increasing the data efficiency of machine learning models, such as transfer learning [35], data augmentation [36], few-shot learning [37], and meta-learning [38]. Many of such methods, however, are not directly transferable to the generative setup. Consequently, it is important to further explore data-efficient approaches for generative models, making them applicable to users' custom and often small-sized data.

1.2.3 *Computational efficiency*

Last but not least, computational efficiency plays an essential role in streamlining the wider adoption of generative models. Everyday users often have access to very limited computational resources for optimization and inference of their models. Even for large technological companies, large-scale deployment of generative models requires meeting an increasing demand by thousands or millions of users around the world, interacting with such models on their personal computers or mobile devices. Computationally demanding models not only restrict accessibility but also significantly increase financial and environmental costs. There exist general methods for the deployment efficiency of deep neural networks, such as model distillation [39], pruning [40], and quantization [41]. However, incorporating task-specific considerations during the design of generative models could further enhance their computational efficiency. Such computational improvements can be explored in both the model architecture, as well as the training and inference strategies.

1.3 FROM 2D TO 3D GENERATIVE MODELING

While advancements in 2D generative models have revolutionized image synthesis and editing tasks, the lack of sufficient 3D priors usually limits their capacity to

represent and generate objects and scenes in 3D. As we live in and interact with a 3D world, the progression from 2D to 3D generative modeling brings us closer to a more immersive experience, unlocking new opportunities in applications such as augmented/virtual reality, gaming, medicine, and architecture. However, transitioning to 3D introduces a new layer of complexity, thereby amplifying challenges related to accessibility.

Firstly, the acquisition and annotation of 3D data, such as multi-view images, meshes, and point clouds, are considerably more challenging. Moreover, 3D generation inherently involves higher dimensionality and modeling geometric structures, which significantly increase the computational demands during both training and inference stages. Furthermore, controllability in 3D generative models also becomes more intricate, as steering the generation in 3D requires additional guidance from users and extra annotations for the training data. Consequently, 3D generative modeling requires further innovations to overcome these obstacles. Among different strategies to do so, designing more computationally efficient 3D representations, as well as 3D-consistent generation using single-view image datasets or pre-trained 2D generative models have shown great promise in recent years. Nevertheless, numerous opportunities remain to be explored in order to unlock the full potential of 3D generative models for real-world applications.

1.4 THESIS OVERVIEW

In this thesis, we aim to identify and address some of the problems arising in existing generative modeling paradigms when used under real-world constraints. We specifically focus on tackling the limitations of controllable generative models (a.k.a conditional generative models), including models conditioned on object categories (classes), textual descriptions, and camera viewpoints. In the next two chapters of the thesis, we address the training of 2D GANs when the training data is small. Particularly, in Chapter 2, we explore a method for efficient transfer learning in class-conditional GANs, and in Chapter 3, we address the training instability of class-conditional GANs under a small data regime. In chapters 4 and 5, we focus on conditional generative models for 3D generation and editing. In particular, in Chapter 4, we investigate the task of 3D-aware image generation from single-view image datasets in applications where 3D data is not available, with a focus on the computational efficiency of the inference. Finally, in Chapter 5, we address the task of text-driven 3D scene editing without the need for supervision with 3D data or any 3D guidance by the users.

1.4.1 *Class-Specific Transfer Learning in GANs*

Transfer learning has been a well-established practice for training deep learning methods on small datasets. The methods common in discriminative models, however, do not directly transfer to the generative setup. Early studies have shown the potential of transfer learning for GANs, mainly focusing on unconditional models. The same, however, has not been well-studied in the case of conditional GANs (cGANs), which provides new opportunities for knowledge transfer compared to unconditional setup. In Chapter 2 of this thesis, we specifically focus on transfer learning in class-conditional GANs. We introduce a new class-specific transfer learning method, called cGANTransfer, to explicitly propagate the knowledge from the old classes to the new ones. The key idea is to enforce the popularly used conditional batch normalization (BN) to learn the class-specific information of the new classes from that of the old ones, with implicit knowledge sharing among the new ones. This enables efficient knowledge propagation from the old classes to the new ones, with a linear increase in BN parameters as the number of new classes grows.

1.4.2 *Transitional Conditioning in GANs*

Class-conditioning in GANs offers a direct means to control the generation based on a discrete input variable. While necessary in many applications, the additional information from class labels could be expected to benefit GAN training. In Chapter 3, on the contrary, we observe that class-conditioning causes mode collapse in limited data settings, where unconditional learning leads to satisfactory generative ability. Motivated by this observation, we propose a training strategy for class-conditional GANs, called transitional-cGAN, that effectively prevents the observed mode-collapse by leveraging unconditional learning. Our training strategy starts with an unconditional GAN and gradually injects the class conditioning into the generator and the objective function. The proposed method for training cGANs with limited data results not only in stable training but also in generating high-quality images, thanks to the early-stage exploitation of the shared information across classes.

1.4.3 *Efficient 3D-Aware Generation with Convolutions*

One of the main applications of 3D generative models is novel view synthesis, where the models can generate arbitrary views of novel scenes, conditioned on a target view. As providing 3D data for generative novel view synthesis is challenging, 3D-aware generative models capable of learning 3D priors from single-view images, such as pose-conditioned GANs, have recently become popular. Naive pose-conditioned

convolutional generative models struggle with generating high-quality, 3D-consistent images from single-view datasets, due to a lack of sufficient 3D priors. The recent integration of Neural Radiance Fields (NeRFs) and generative models, such as GANs, has transformed 3D-aware generation from single-view images. NeRF-GANs exploit the strong inductive bias of neural 3D representations and volumetric rendering, at the cost of higher computational complexity. In Chapter 4, we aim to revisit pose-conditioned 2D GANs for efficient 3D-aware generation at inference time by distilling 3D knowledge from pretrained NeRF-GANs. We propose a simple and effective method for efficient inference of 3D-aware GANs, based on re-using the well-disentangled latent space of a pre-trained NeRF-GAN in a pose-conditioned convolutional network, to directly generate 3D-consistent images corresponding to the underlying 3D representations.

1.4.4 *Generative 3D Object Insertion*

3D generation and editing become even more challenging when it comes to modeling complex and compositional 3D scenes. Therefore, several studies have recently exploited the strong priors of the recent language-based 2D diffusion models for the task of 3D generation and editing. 2D diffusion models have been used for 3D asset generation [42], single-image object reconstruction [43], and 3D scene editing [44]. In the particular case of 3D scene editing, the existing methods are mainly limited to modifying [44–46] or removing [47, 48] existing objects in the scenes. Generating new objects, however, remains a challenge for such methods. In Chapter 5, we tackle the task of object generation in 3D scenes without the need for any 3D supervision or 3D placement guidance from the users. We introduce InseRF, a novel method for generative object insertion in the NeRF reconstructions of 3D scenes. Based on a user-provided textual description and only a 2D bounding box in a reference viewpoint, InseRF generates new objects in 3D scenes. Specifically, we propose grounding the 3D object insertion to a 2D object insertion in a reference view of the scene. The 2D edit is then lifted to 3D using a single-view object reconstruction method. The reconstructed object is then inserted into the scene, guided by the priors of monocular depth estimation methods. InseRF is capable of controllable and 3D-consistent object insertion without requiring explicit 3D information as input.

2

CLASS-SPECIFIC TRANSFER LEARNING IN GANS

Generative adversarial networks (GANs) have shown impressive results in both unconditional and conditional image generation. Previous studies have shown that pre-trained GANs on different datasets can be transferred to improve the image generation from a small target dataset. The same, however, has not been well-studied in the case of conditional GANs (cGANs), which provides new opportunities for knowledge transfer compared to the unconditional setup. In particular, the new classes may borrow knowledge from the related old classes, or share knowledge among themselves to improve the training. This motivates us to study the problem of efficient conditional GAN transfer with knowledge propagation across classes. To address this problem, we introduce a new GAN transfer method to explicitly propagate knowledge from the old classes to the new classes. The key idea is to enforce the popularly used conditional batch normalization (BN) to learn the class-specific information of the new classes from the old classes, with implicit knowledge sharing among the new ones. This allows for an efficient knowledge propagation from the old classes to the new ones, with the BN parameters increasing linearly with the number of new classes. The extensive evaluation demonstrates the clear superiority of the proposed method over state-of-the-art competitors for efficient conditional GAN transfer tasks.

2.1 INTRODUCTION

Generative adversarial networks (GANs) [15, 49] are the most common models used for image and video generation [50], showing very promising results in unconditional [19, 51, 52] and conditional [18, 53, 54] setups.

Learning from limited data is a well-studied problem in the discriminative setup, where the concept of knowledge transfer [55] between two different but related tasks [56] or domains [57] is ubiquitous. In contrast, literature on transfer learning for generative adversarial models is fairly limited. One may find this unexpected, since many popular knowledge transfer methods in discriminative setup, in turn, use generative schemes [58, 59]. However, the limited literature is less surprising when the complexity of adversarial training and the mode collapse are taken into account.

A notable work by *Wang et al.* [60] first addressed the problem of training GANs on limited data using a careful fine-tuning (FT) strategy. Follow-up works [61–63]

are the variants of [60] that focus on better fine-tuning strategies. On the contrary, *Noguchi & Harada* [64] proposed the batch statistics adaptation (BSA) technique, by learning only the batch normalization parameters on a small target dataset. As most of the previous works [60, 62, 64] primarily focus on the case of unconditional GANs, we investigate in a different direction of conditional GANs (cGANs). In particular, we are interested in producing new classes given a pre-trained class-conditional GAN. cGANs are strikingly interesting due to their capability of handling a large number of classes with a single network. For example, *BigGAN* [18] can generate images from all 1K classes of ImageNet [65]. In fact, *BigGAN* is used as the pre-trained network even by unconditional methods [62, 64]. We refrain from fine-tuning whenever possible, as we believe that new classes can be introduced within such powerful cGANs. Moreover, some powerful pre-trained cGANs can potentially be used to add new classes in the lifelong learning [64, 66–68] fashion, which however is beyond the focus of our paper.

In this work, we study how new classes with a limited amount of samples can be added to pre-trained cGANs using knowledge transfer across classes. To do so, inspired by BSA [64], we aim at learning only the batch normalization (BN) parameters that generally encode the class-specific information. Our key idea, different from [64], relies on the assumption that the knowledge between old and new classes can also be transferred by searching for the similarity between them. Our experimental setup, however, does not allow us to access the old data used for the pre-trained model. Therefore, the similarity is searched in the conditional space of the BN parameters, during the training of cGANs. In this process, we learn the similarity scores *explicitly between old and new classes, and implicitly between new ones*. The learned old-to-new similarity scores are then used to derive the batch statistics of new classes from that of old ones.

It is well-established in [66, 69–74] and many other works, that learning algorithms can greatly benefit from the shared knowledge between classes. Often, such similarity is either known or discovered when all the classes are accessible. In the context of domain generalization (or in some special case of adaptation), the source data is similarly inaccessible partially or completely [75–77]. However, the latter assumes that the new classes are either the same or largely overlap with the old ones. Note that in our case, *new classes do not even overlap with the inaccessible old ones*. In addition, almost all aforementioned works seek similarity in the feature space with an exception of [74]. However, [74] is primarily designed to serve discriminative models. Our generative case, on the other hand, hinders us to access the feature space. We, therefore, rely on the conditional space of cGANs to establish the sought class similarities. Up to our knowledge, we learn the inter-class similarities in the conditional space of the generative models, for the first time.

In summary, we utilize cross-class knowledge while introducing new classes in cGANs. While doing so, active searching of similarity scores between new and old classes with implicit knowledge sharing among new ones is suggested. In this context, we propose a novel method for finding the similarity between new and old classes without requiring access to the old data. The proposed method is particularly suitable when transferring knowledge from pre-trained cGANs.

In summary, the key contributions of our work are as follows:

- We study the new problem of efficient GAN transfer to new classes with explicit inter-class knowledge propagation in pre-trained cGANs.
- We propose a novel method for learning similarity between old and new classes and knowledge sharing between new classes using the batch normalization statistics of the old classes in the conditional space of GANs.
- Our experiments on three datasets demonstrate the superiority of our method both in terms of generated image quality and the convergence speed.

2.2 RELATED WORK

Class-conditional GANs: Different architectures and loss functions have been proposed for conditioning GANs on class labels [78–80]. The current state-of-the-art methods for class conditioning commonly employ cGAN with projection discriminator [79, 81]. In the generator, conditional batch normalization [82] with class-specific scale and shift parameters are applied to each layer of the generator. The discriminator is conditioned on the class labels by computing the dot product of the last feature layer and the learnable embedding of the desired class. The performance of the conditional GANs was further improved by adding self-attention layers to the generator and the discriminator [83]. BigGAN [18] was able to reach state-of-the-art performance on image generation from ImageNet, mainly by using a bigger batch size and some architectural improvements such as a hierarchical latent variable. The conditioning, however, still happens through the class-conditional batch normalization in the generator and the projection layer in the discriminator.

Transfer Learning in GANs: Iterative image generation approaches, such as DGN-AM [84] and PPGN [85], could be considered as early attempts at transfer learning in image generation by generating images via maximizing the activation of the neurons of a pre-trained classifier. TransferGAN [60] is one of the earliest studies addressing transfer learning in GANs. The authors showed that, by simply fine-tuning a pre-trained network on the target dataset, they can outperform training from scratch in terms of image quality and convergence time. However, naive fine-tuning on small data still suffers from mode collapse and training instability. Another method [63] proposes transferring the low-level layers of the generator and the

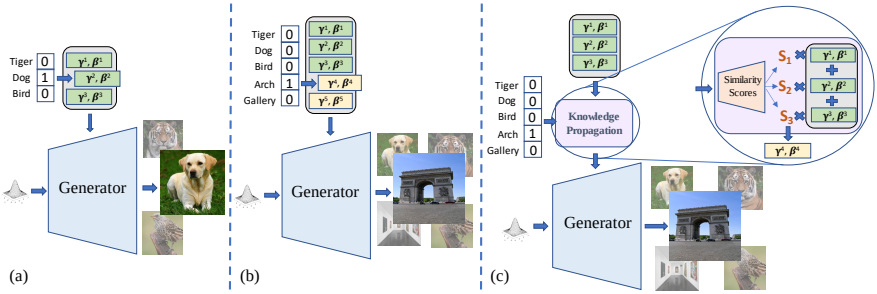


FIGURE 2.1: (a): A generator trained on a multi-class dataset. (b): The generator, extended by new classes without taking the old classes into account. (c): The conceptual diagram of our proposed method for conditional GAN transfer with knowledge propagation across classes. Here γ^i and β^i $(1 \leq i \leq 5)$ represent the class-specific parameters.

discriminator from the pre-trained network while learning the high-level layers from scratch for the target data. In a recent study [61], it is shown that simply freezing the low-level filters of the discriminator is more effective than previous fine-tuning approaches. BSA [64], on the other hand, instead of seeking ways of fine-tuning the network, proposed freezing the weights of the pre-trained generator except the batch normalization parameters. For the target data, new BN parameters are learned without fine-tuning the generator. This allows BSA to add new classes without disturbing the old ones. MineGAN [62] learns a small fully-connected miner network at the input of a frozen pre-trained GAN. The miner learns to shift the prior input distribution to the most suitable one for the target data. After training the miner, MineGAN further fine-tunes both the generator and the miner as the final model. MineGAN is designed to transfer knowledge to a single-class target.

Inter-class Knowledge Transfer: Many works have emerged that focus on extending models trained on previous examples/images to perform well on new data, and here is where knowledge transfer becomes essential. For instance, [71] uses memory and attention modules to transfer labeled data knowledge to a new class example. A closely related work [73] uses the class similarity between source classes and a target class to improve the classification performance. [72] is a few-shot meta-learning method that uses the so-called relation score of a new class example with previous examples in order to classify the new example. In a more related work [74], BN layers have been used for knowledge transfer from old classes, mainly targeting binary classification with a brief empirical study on style transfer. The knowledge transfer framework of [74] separately performs encoding, pre-selection, and combination of

old classes. In contrast, our method jointly optimizes the prior knowledge and the transfer process, leading to an efficient learning paradigm for generative models.

2.3 PROBLEM DEFINITION

The task of class-conditional GAN transfer aims at approximating the target data distribution by transferring knowledge from the source multi-class data to the target data using pre-trained GAN models. As shown in Fig. 2.1 (a), the pre-trained GAN model consists of one generator G and one discriminator D, jointly trained on the source multi-class dataset $\mathcal{X} = \{X^1, X^2, \ldots, X^N\}$, in which X^y is a set of real images in category $y \in \{1, 2, \ldots, N\}$ with the underlying distribution $P_y(x) \in \{P_1(x), P_2(x), \ldots, P_N(x)\}$. In the conditional setup, given a random noise vector z (usually sampled from $\mathcal{N}(0, I)$) and a class label y as inputs, G is trained in an adversarial game with D, to generate an image $x \sim P_y(x)$:

$$\begin{aligned} &x = G(z, y); \\ &z \sim \mathcal{N}(0, I),\ y \in \{1, \ldots, N\} \quad \text{s.t. } x \sim P_y(x). \end{aligned} \tag{2.1}$$

In the state-of-the-art conditional GANs (e. g., [18, 79, 83]), G is commonly conditioned on the class labels using conditional batch normalization. Specifically, the layer-wise output f_l, of layer $l \in \{1, \ldots, L\}$, is normalized and modulated by the class-specific scale $\gamma_l^y \in \{\gamma_l^1, \ldots, \gamma_l^N\}$ and shift $\beta_l^y \in \{\beta_l^1, \ldots, \beta_l^N\}$ as:

$$f_l' = \gamma_l^y \frac{f_l - \mu_l}{\sigma_l} + \beta_l^y, \tag{2.2}$$

where μ_l and σ_l represent the batch mean and variance for the l-th layer, and f_l' is the normalized output of the layer. Thus, the class-specific information is parameterized by the corresponding scales and shifts.

Transfer learning in conditional GANs can be defined as exploiting pre-trained GAN models to adapt the generator/discriminator to a new multi-class target data $\mathcal{X}' = \{X^{N+1}, X^{N+2}, \ldots, X^{N+M}\}$ with M new categories $\{y'\}$, $y' \in \{N+1, \ldots, N+M\}$. Mathematically, this task can be defined as the following learning problem:

$$\begin{aligned} &x' = G'(z, y'); \\ &z \sim \mathcal{N}(0, I),\ y' \in \{N+1, ..., N+M\} \\ &\text{s.t. } x' \sim P_{y'}(x),\ \text{ given } G(z, y), \end{aligned} \tag{2.3}$$

where G' is the new generator learned for the new categories $\{y'\}$. It is often desirable that G' can also generate the previous classes, as shown in Fig. 2.1 (b).

The problem of learning such a generator which is capable of generating images for both $\{y\}$ and $\{y'\}$ can be formulated as:

$$\begin{aligned} & x = G'(z, y_f); \\ & z \sim \mathcal{N}(0, I),\ y_f \in \{1, ..., N+M\} \\ & \text{s.t. } x \sim P_{y_f}(x),\ \text{ given } G(z, y), \end{aligned} \tag{2.4}$$

where we denote the final category set as $\{y_f\} = \{y\} \cup \{y'\}$ and the extended generator as G'.

Transfer learning in the context of GANs is usually approached by a careful fine-tuning of a pre-trained model [60–63], with or without modifying the architecture. These architecture modifications include adding either new layers or new batch normalization parameters, to additionally learn new knowledge of the target data. Such approaches overlook the shared similarities among the old and new classes, resulting in an inefficient knowledge propagation. This motivates us to aim for a more efficient conditional GAN transfer. In particular, we seek explicit knowledge propagation from old (*i.e.* source) classes to the new (*i.e.* target) ones, along with the possibility of knowledge sharing among target classes. Fig. 2.1 (c) conceptually illustrates the problem addressed in this paper.

2.4 KNOWLEDGE TRANSFER ACROSS CLASSES

We first provide an overview of the proposed method, followed by the details. Our premise is built upon a known observation in BigGAN [18], where interpolating between different classes produces *visually meaningful* intermediate images that do not exist in the training data. This implies that the learned class-specific parameters could lie on a smooth manifold, on which new classes also reside. Consequently, it begs the question: can the similarities between the source and the target classes–as well as those within the target–be exploited to learn the target representations? This question leads us to learn the parameters of the new classes, while being dependent upon the parameters (representing the knowledge) of old classes. More precisely, we propose to obtain the representation of each target class by learning a suitable linear combination over the representations of the source classes – which we call *knowledge propagation*. In addition, we propose a mechanism to enable *knowledge sharing* within target classes by optimizing the source knowledge in favor of the multiple target classes[1]. We address the exact problem defined by eq. (2.4), where the model aims to generate both new and old classes. Such consideration is often

1 Although the focus of our work is on the multi-class target data, our method can also be generalized effectively to single-class targets (please refer to appendix A).

ignored in the literature, with an exception of BSA [64]. The details of the proposed knowledge propagation and sharing strategies are presented in Secs. 2.4.1 and 2.4.2.

2.4.1 *Knowledge Propagation*

Following the class-conditioning paradigm discussed in Sec. 2.3, we embed the class-specific representations in the batch normalization (BN) layers as the scale and shift parameters. Accordingly, knowledge propagation is performed in BN layers. To obtain the BN parameters (γ_l and β_l in eq. (2.2)) of each new class, our model linearly combines the BN parameters of old classes $\{y\}$ using layer-wise similarity scores, learned during the training step:

$$\begin{aligned} \gamma_l^{y'} &= [\gamma_l^1, \gamma_l^2, \ldots, \gamma_l^N] S_{\gamma_l}^{y'}, \\ \beta_l^{y'} &= [\beta_l^1, \beta_l^2, \ldots, \beta_l^N] S_{\beta_l}^{y'}, \\ y' &\in \{N+1, \ldots, N+M\}, \;\; l \in \{1, \ldots, L\}. \end{aligned} \tag{2.5}$$

Here, $S_{\gamma_l}^{y'} = [s_{\gamma_l}^{(y',1)}, \ldots, s_{\gamma_l}^{(y',N)}]^\top \in \mathbb{R}^N$ and similarly $S_{\beta_l}^{y'} \in \mathbb{R}^N$ are vectors of learned scores for the class y' in layer l. Two things are to be noted here. First, we learn the similarity scores for scale and shift parameters separately. This is because, the new classes could be similar to some of the old classes in terms of their distribution mean, while being similar to another set of classes in terms of intra-class variance. Secondly, we also learn a different set of scores per layer, since different layers of the network do not necessarily benefit from the same set of old classes. It is well-known that different layers of neural networks represent different levels of feature representation. As intuitively shown in Fig. 2.2, some layers could be responsible for the general shape, some for the color and the texture, and some for finer details. Such hierarchy of features is also the main motivation of StyleGAN [19] for using layer-specific styles.

Based on the empirical observations, we propagate class-specific knowledge only in the generator while simply fine-tune the discriminator. A similar knowledge propagation in the discriminator leads to performance degradation in our experiments. One possible reason for such degradation is that the knowledge propagation speeds up the over-fitting problem of the discriminator on small datasets [87].

2.4.2 *Knowledge Sharing*

In addition to knowledge propagation, we propose a mechanism for knowledge sharing among target classes. Consider transferring a conditional GAN, pre-trained

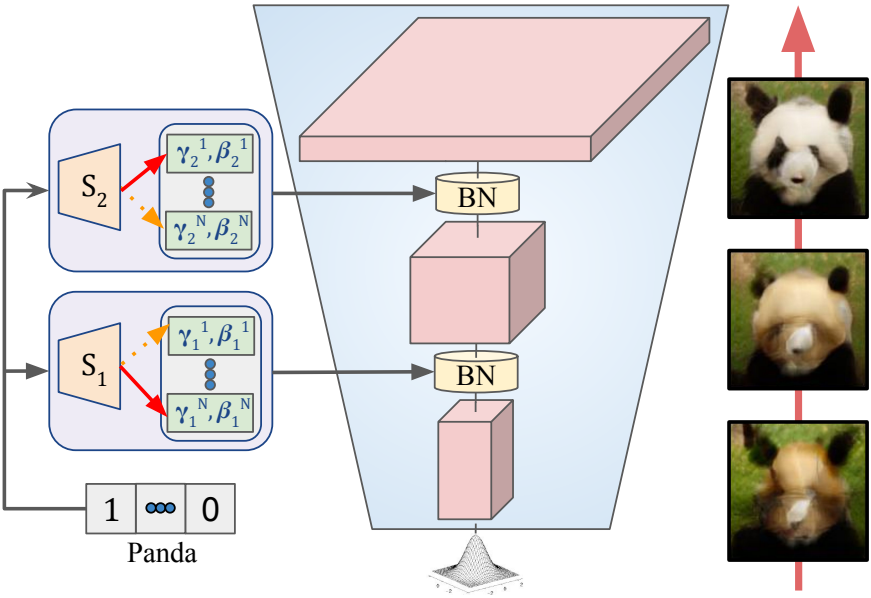


FIGURE 2.2: Intuitive visualization of how different layers might borrow information from different classes based on the hierarchy of features such as shape and color (for a better visualization, only two layers are illustrated). Pictures in the figure are obtained from our experiment on AnimalFace dataset [86] (Please refer to Fig. 2.9 for more visualizations).

on ImageNet classes, to the AnimalFace [86] dataset containing 20 classes of different animal faces. Although faces of different animals have their own unique characteristics, they still share a level of common structure. Therefore, instead of finding each class representation independently, it is reasonable to exploit the shared knowledge between all target classes. A basic but indirect example of knowledge sharing between the target classes can be seen in methods based on fine-tuning the pre-trained convolutional filters, where all target classes contribute to optimizing the same filters. The same, however, does not happen when learning the class-specific BN parameters independent of other classes.

To empower the knowledge sharing among target classes in our method, we propose to allow the target classes to jointly optimize the prior knowledge (pre-training BN parameters) during knowledge propagation. Optimizing the prior knowledge will enable the model to obtain a shared set of intermediate representations, which are more suitable for all target classes. These shared representations–which we name

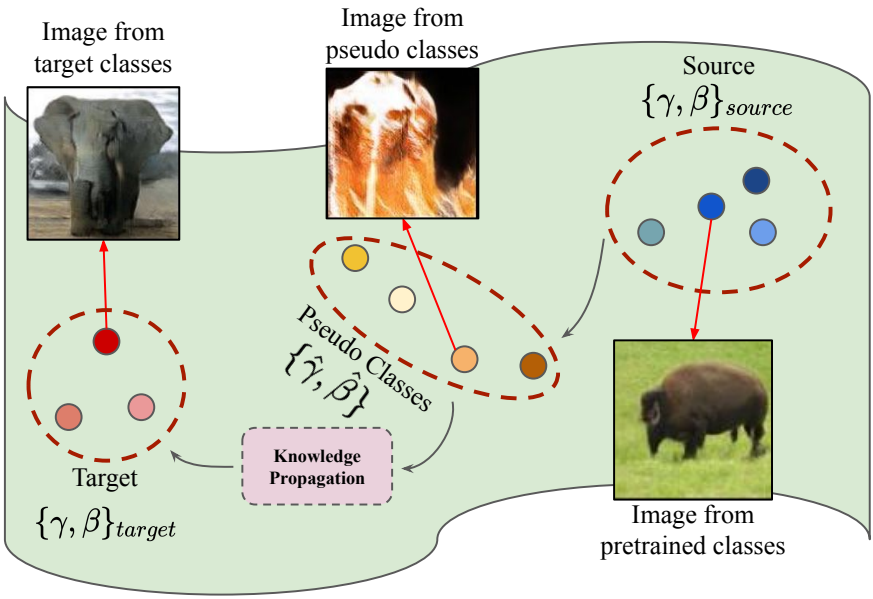


FIGURE 2.3: The visualization of the proposed knowledge sharing and propagation. The sharing takes place in obtaining the pseudo-classes, by updating the source. The propagation step combines pseudo-classes to obtain the target. The sharing and the propagation are performed jointly. The images are obtained from our experiments (Please refer to Figs. 2.7 and 2.8).

pseudo-classes–then can be combined according to the similarity scores during the knowledge propagation step. Mathematically, we rewrite the propagation equation after knowledge sharing as:

$$\begin{aligned} \gamma_l^{y'} &= [\hat{\gamma}_l^1, \hat{\gamma}_l^2, \ldots, \hat{\gamma}_l^N] S_{\gamma_l}^{y'}, \\ \beta_l^{y'} &= [\hat{\beta}_l^1, \hat{\beta}_l^2, \ldots, \hat{\beta}_l^N] S_{\beta_l}^{y'}, \\ y' &\in \{N+1, \ldots, N+M\}, \;\; l \in \{1, \ldots, l\}. \end{aligned} \tag{2.6}$$

where $[\hat{\gamma}_l^y]$ and $[\hat{\beta}_l^y]$ are obtained by updating a copy of the source BN parameters, forming the shared representations for layer l. It is important to note that the modified BN parameters in eq. (2.6) are used only for learning the new classes. We do not replace the original old BN parameters, as we want to preserve the old knowledge during the transfer. This process does not further increase the number of parameters for inference, since the updated parameters (and the similarity scores) are discarded

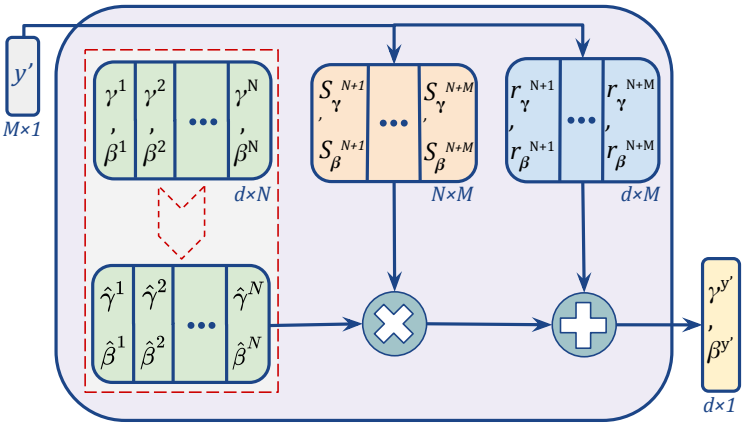


FIGURE 2.4: An overview of the proposed block for cGAN transfer with knowledge propagation across classes.

after learning the target BN parameters. Fig. 2.3 visualizes the proposed approach for knowledge sharing in conjunction with knowledge propagation.

2.4.3 *Training with Residuals and Sparsity*

For the purpose of generalizability, we also consider the classes that cannot be represented well only by combining the shared intermediate representations. To address this issue, we propose to add residual vectors $r^{y'}_{\gamma_l}$ and $r^{y'}_{\beta_l}$ to the scale and shift parameters obtained from knowledge propagation, respectively. However, in order to encourage the model to use the prior knowledge as much as possible, we minimize the magnitude of the residual vectors using ℓ_2 regularization. Moreover, to encourage the new classes to learn from the most relevant prior knowledge, we also add an ℓ_1 sparsity regularization on the similarity scores.

In summary, we propose conditional GAN transfer from a pre-trained GAN to multi-class target data via BN parameters of pre-training classes, which are linearly combined to obtain the new classes' representations. Moreover, we enable sharing knowledge within the target classes by allowing them to update the prior knowledge according to the needs of the target data. An overview of the complete proposed method is illustrated in Fig. 2.4.

The final loss function of our method follows eq. (2.7), where L_g is the adversarial loss [15, 49, 88] used for training the GAN, L_r is an ℓ_2 regularization over the residuals, and L_s is an ℓ_1 regularization over the similarity scores:

$$\begin{aligned} L &= L_g + \lambda_r L_r + \lambda_s L_s, \\ &= L_g + \lambda_r \sum_{l=1}^{L} \sum_{y'=N+1}^{N+M} \{||r_{\gamma_l}^{y'}||_2^2 + ||r_{\beta_l}^{y'}||_2^2\} \\ &\quad + \lambda_s \sum_{l=1}^{L} \sum_{y'=N+1}^{N+M} \{|S_{\gamma_l}^{y'}| + |S_{\beta_l}^{y'}|\}, \end{aligned} \tag{2.7}$$

where λ_r and λ_s are the hyper-parameters associated to residual and sparsity losses, respectively.

2.5 EXPERIMENTS

In this section, we provide the details of our experiments, the evaluation of our method (cGANTransfer), and its comparison with the baseline methods in two different setups. Then, we provide further analysis of our contributions with an ablation study and explanatory visualizations.

2.5.1 *Experimental Setup*

Datasets: To evaluate our method, we use two main experimental setups. In the first setup, we pre-train the network on 80 randomly-selected classes of CIFAR100 [89]. The remaining classes are used as the target. To have a more thorough evaluation, we evaluate our method on CIFAR100 with different numbers of target classes and images per class. For the second setup, we consider the more challenging task of extending a network pre-trained on ImageNet [65] to the Places365 dataset [90]. Following MineGAN [62], we select 5 classes (i.e., Alley, Arch, Art Gallery, Auditorium, Ballroom) and down-sample each class to 500 images. In addition, we use AnimalFace dataset [86]–containing 20 classes–for further analysis and visualizations. We down-sample each class to contain a maximum of 100 images.
Architecture: For the cGAN architecture, we use BigGAN [18] with hinge loss, as it is one of the most widely-used state-of-the-art cGANs. We use hierarchical noise for training on ImageNet, but not for CIFAR100, following the experimental setup of BigGAN [18].
Training: To conduct our experiments in the class extension setup, we freeze the weights of the generator and learn only the parameters of our knowledge transfer

Method/ Experiment	20/ 600	20/ 300	20/ 100	10/ 600	10/ 300	10/ 100	mFID
Scratch	**25.35**	47.20	81.90	52.23	62.17	83.68	58.76
TransferGAN	27.67	31.19	48.67	30.58	40.74	64.75	40.60
BSA - early	37.12	32.10	50.88	42.01	40.34	66.76	44.87
BSA - full	28.03	**30.89**	40.73	30.73	35.62	56.48	37.08
Ours	29.10	30.99	**40.04**	**29.95**	**35.23**	**54.95**	**36.71**

TABLE 2.1: Results of the evaluation and the comparison of the proposed method using FID and KMMD scores for transferring 80 classes of CIFAR100 to 20 classes or 10 classes. Different columns report the best FID scores in different cases. The format A/B means using A classes and B images per class for the target. BSA-early indicates the FID scores for BSA at the iteration where our method achieves its best FID score.

block. We also find that, if the hierarchical noise is used (e.g. ImageNet setup), it is also necessary to fine-tune the linear layers that project the noise into the BN parameters' space. The reason might be that these linear layers are optimized to add the detailed style of the dataset to the generated images. Experiments on CIFAR100 are conducted using a single V100 GPU with a batch size of 50, and the experiments on the second setup are performed using 8 V100 GPUs with a batch size of 256.

2.5.2 *Comparison with the State-of-the-Art*

To quantitatively evaluate our method, we use Fréchet Inception Distance (FID) [91]. We also provide the KMMD metric (Gaussian kernel with $\sigma = 1$) for ImageNet setup, following BSA and MineGAN [62, 64]. We compare our method against training from scratch (Scratch), TransferGAN [60], PPGN [85], MineGAN [62] and BSA [64]. Among these methods, BSA is the only one that performs the class extension. Therefore, we consider BSA as our main baseline.

Tab. 2.1 shows the results for evaluating our method on CIFAR100 and its comparison with the other methods. The experiments include two different numbers of classes (20 and 10), and 3 different numbers of samples per class (600, 300, 100). As can be seen, decreasing the number of training samples degrades the performance of learning from scratch and TransferGAN significantly. On the other hand, BSA and our method perform more robustly on small data. Compared to BSA, our method achieves slightly better FID scores (possibly because of the less challenging task on CIFAR) and a significantly better convergence speed (more than **4x** speed-up on average), as visualized in Fig. 2.5 (left). Our method enjoys such speed-up thanks

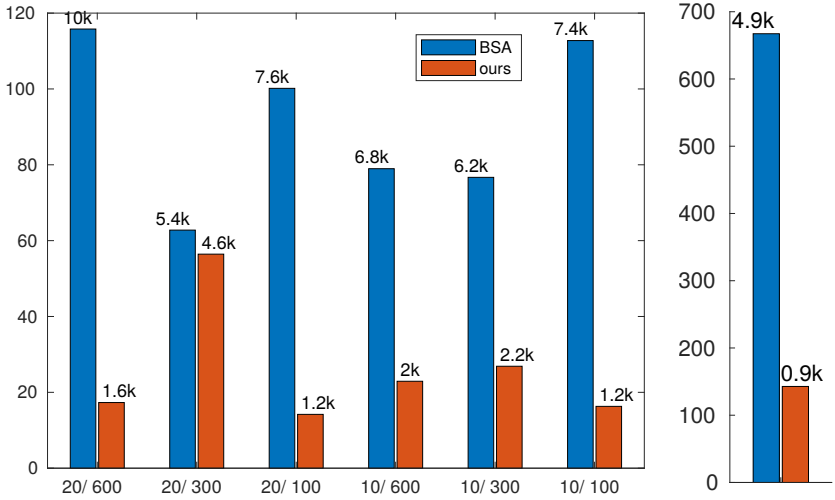


FIGURE 2.5: The convergence comparison between BSA and ours on CIFAR (left) and Places365 (right) setups. Note that the number of iterations is shown above each bar.

to the knowledge propagation through the class representations, as opposed to its counterpart–BSA.

The superiority of our method becomes even more noticeable in the more challenging setup of transferring ImageNet to Places365. The classes of Places365 have large intra-class diversity, and there is a considerable domain gap between the source and target datasets. Therefore, the task becomes highly challenging when only 500 images per class are used. Tab. 2.2 presents the results for transferring ImageNet to 5 classes of Places365. As can be seen, our method clearly outperforms the other methods in terms of FID and KMMD. Fig. 2.5 (right) again shows that the proposed method is more than **5x** faster than BSA in terms of convergence speed. The comparison between our method and BSA shows that knowledge transfer across classes results in not only faster convergence but also a more accurate distribution approximation of the target data. Fig. 2.6 depicts the visual results for our method and the compared ones in ImageNet-to-Places365 setup.

Experiment	FID	KMMD
Scratch*	190	0.96
TransferGAN*	89.2	0.53
PPGN*	139	0.56
MineGAN*	82.3	0.47
BSA	85.9	0.18
Ours	**71.1**	**0.16**

TABLE 2.2: Results of the evaluation and the comparison of the proposed method using FID and KMMD scores for transferring ImageNet to 5 classes of Places365 with 500 samples per class. * adopted from [62].

2.5.3 *Ablation Study*

To show the effect of each component in our method, we perform the following experiments on ImageNet-to-Places365 setup: Transferring from previous classes without updating the prior knowledge or residual learning, transferring from updatable prior knowledge without residual learning, updatable prior with residuals, sharing the combination weights between the layers, and the experiments on the regularizations. Tab. 2.3 shows the results for the ablation study of our method when transferring from ImageNet to Places. The ablation study shows that, all the proposed components contribute meaningfully to the final model.

2.5.4 *Fine-tuning*

As mentioned in Sec. 2.5.1, we freeze the weights of the generator in our experiments. This enables a parameter-efficient extension of the cGAN to target classes. However, when parameter efficiency is not a constraint, fine-tuning the filters could further improve the performance of the model, as shown in [62]. Therefore, we also provide the results of further fine-tuning of our model after learning the target BN parameters in Tab. 2.4. To fine-tune the model, we freeze the knowledge propagation parameters (prior knowledge and similarity scores) and only fine-tune the residuals, as well as the rest of the model. The results show the additional benefit of further fine-tuning of our model.

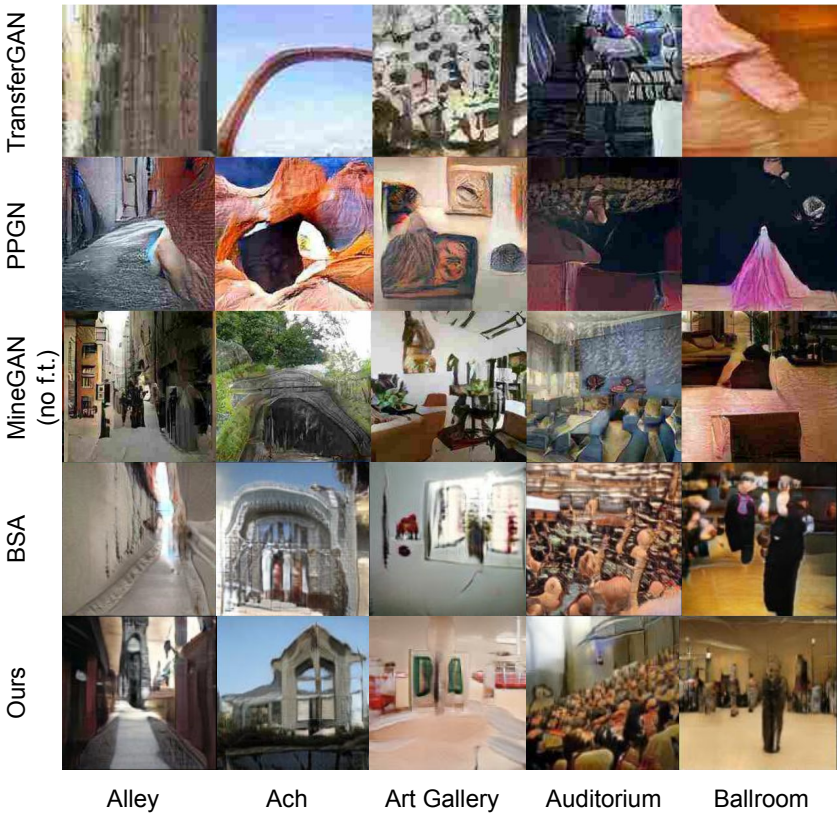


FIGURE 2.6: The visual comparison between our proposed method and baseline methods on Places365. The first three competitors' results are adopted from [62].

2.5.5 *Analysis and Visualizations*

To better understand the proposed method, we further analyze its main components while transferring ImageNet to AnimalFace. The quantitative (FID, KMMD, and convergence time) and qualitative results of class extension to AnimalFace, with and without class knowledge propagation, are shown in Fig. 2.7. The results for AnimalFace are consistent with the previous results provided in Sec. 2.5.2.

Knowledge sharing within target classes: An important component of our method is knowledge sharing within the target classes. Fig. 2.8. shows the pseudo-classes (intermediate representations) obtained by our method after training on the target

Experiment	Best FID	Iterations
Freezed Prior, w/o Res	96.3	3000
Freezed Prior, with Res	79.4	1100
Tunable Prior, w/o Res	81.2	4000
Shared W, Tunable Prior, w/o Res	79.8	3400
Final architecture w/o reg	82.4	2400
Final architecture w/o l1	84.7	1600
Final architecture w/o l2	81.1	2100
Final architecture, with l1 & l2	**71.1**	**900**

TABLE 2.3: Ablation study over GAN transfer from ImageNet to Places365. Iterations: iteration number for the best FID.

Dataset	FID	KMMD
Places365	65.48	0.156
AnimalFace	68.92	0.194

TABLE 2.4: The results of further fine-tuning of our model on Places365 and AnimalFace.

data. To visualize these representations, we directly sample the BN parameters from the updated base representations, instead of combining them to obtain the new class representations (refer to Fig. 2.3). As it can be seen, the initial class representations of the ImageNet are transformed to some intermediate “pseudo-class" representations that contain the shared structure of the target faces, but do not belong to a particular target class. Thus, our method is able to obtain the representations of each new class by combining these new pseudo-classes, which are learned using all of the target classes.

Layer-specific knowledge transfer: Another key aspect of our method, as discussed in Sec. 5.3 (refer to Fig. 2.2) and supported by the ablation study (Tab. 2.3), is that the linear transfer weights are different for each layer. Fig. 2.9 visualizes the effect of using different weights per layer. To obtain these visualizations, after training the network on the target, we use the learned similarity scores of the first layer for all the layers. Then, we gradually introduce the learned scores of the next layers. From the results, we see that the first layers are mostly responsible for the general object shape, and the later layers introduce color and finer details.

Experiment	FID	KMMD	Iters	Time (Mins)
BSA	91.9	0.25	4800	650
Ours	**85.9**	**0.23**	**1400**	**235**

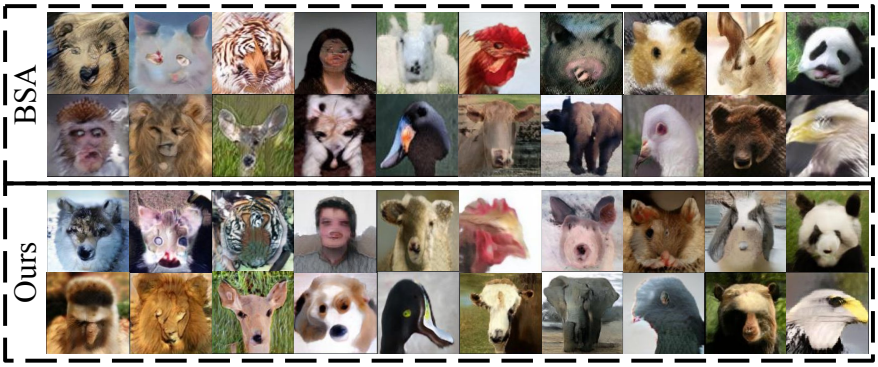


FIGURE 2.7: Quantitative (top) and qualitative (bottom) results of transferring ImageNet to 20 classes of AnimalFace without (BSA) and with (Ours) **cross-class knowledge propagation**. Iters: training iteration number for the best FID.

FIGURE 2.8: **Knowledge sharing.** The intermediate shared representations learned for AnimalFace as the target data.

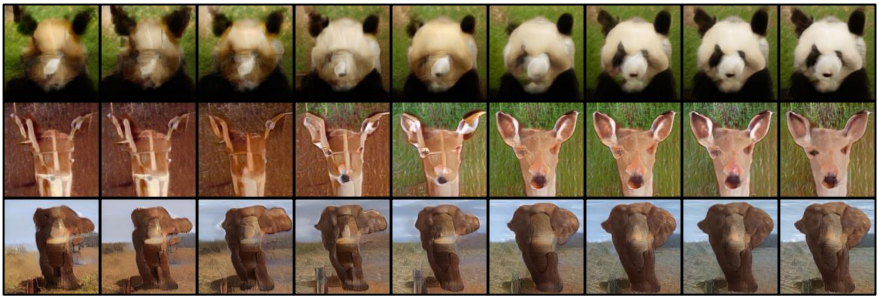


FIGURE 2.9: **Layer-wise knowledge transfer.** Different layers introduce different information to the generated images. The shape is mostly changed by the starting layers, while colors and finer details are added by the later layers.

2.6 CONCLUSION AND FUTURE WORK

In this chapter, we explored the problem of conditional GAN transfer by transferring the knowledge across both source and target classes. We represented the knowledge of individual classes by their respective batch normalization parameters, which are used for conditioning during the generation. To propagate the knowledge to new classes, we introduced a method that learns to update and combine the batch normalization parameters of the source classes. The evaluations on three standard benchmarks demonstrate a clear advantage of our method, both in terms of training efficiency and the image generation quality (measured by FID and KMMD), compared to the state-of-the-art methods. Our ablation study highlighted the importance of jointly using the update and combination steps, which we referred to as sharing and propagation, respectively.

We believe our study can be followed by several interesting future works, such as further knowledge propagation on the discriminator, its integration with differentiable augmentation [25, 87], and the extension to lifelong learning.

3

TRANSITIONAL CONDITIONING IN GANS

Class-conditioning offers a direct means to control a Generative Adversarial Network (GAN) based on a discrete input variable. While necessary in many applications, the additional information provided by the class labels could even be expected to benefit the training of the GAN itself. On the contrary, we observe that class-conditioning causes mode collapse in limited data settings, where unconditional learning leads to satisfactory generative ability. Motivated by this observation, we propose a training strategy for class-conditional GANs (cGANs) that effectively prevents the observed mode-collapse by leveraging unconditional learning. Our training strategy starts with an unconditional GAN and gradually injects the class conditioning into the generator and the objective function. The proposed method for training cGANs with limited data results not only in stable training but also in generating high-quality images, thanks to the early-stage exploitation of the shared information across classes. We analyze the observed mode collapse problem in comprehensive experiments on four datasets. Our approach demonstrates outstanding results compared with state-of-the-art methods and established baselines.

3.1 INTRODUCTION

Since the introduction of generative adversarial networks (GANs) [15], there has been substantial progress in realistic image and video generation. The generated contents are often controlled by conditioning the process by means of conditional GANs [92]. In practice, conditional GANs are of high interest, as they can generate and control a variety of outputs using a single model. Some example applications of conditional GANs include class-conditioned generation [18], image manipulation [93], image-to-image translation [22], and text-to-image generation [94].

Despite the remarkable success, training conditional GANs requires large training data, including conditioning labels, for realistic generation and stable training [95]. Collecting large enough data is challenging in many frequent scenarios, due to the privacy, quality, and diversity required, among other reasons. This difficulty is often worsened further for datasets for conditional training, where also labels need to be collected. The case of fine-grained conditioning adds an additional challenge for data collection, since the availability of the data samples and their variability are expected to deteriorate with the increasingly fine-grained details [96].

While training GANs with limited data has recently received some attention [25, 60, 95], the influence of conditioning in this setting remains unexplored. Compared to the unconditional case, the conditional information provides additional supervision and input to the generator. Intuitively, this additional information could guide the generation process better and ensure the success of conditional GANs whenever its unconditional counterpart succeeds. In fact, one may even argue the additional supervision by conditioning could alleviate the problem of limited data, to an extent. Surprisingly, however, we observe the opposite in our experiments for class-conditional GANs. As visualized in Fig. 3.1, the class-conditional GAN trained on limited data suffers from severe mode collapse. Its unconditional counterpart, on the other hand, trained on the same data, is able to generate diverse images of high fidelity with a stable training process. To our knowledge, these counter-intuitive observations of class-conditional GANs have not been observed or reported before.

In this paper, we first study the behavior of a state-of-the-art class-conditional GAN, by varying the number of classes and image samples per class, and contrast it to the unconditional case. Our study in the limited data regime reveals that the unconditional GANs compare favorably with conditional ones in terms of the generation quality. We, however, are interested in the conditional case, so as to be able to control the image generation process using a single generative model. In this work, we set out to mitigate the aforementioned mode collapse problem.

Motivated by our empirical observations, we propose a method for training class-conditional GANs that leverages the stable training of the unconditional GANs. During the training process, we integrate a gradual transition from unconditional to conditional generative learning. The early stage of the proposed training method favors the unconditional objective for the sake of stability, whereas the later stage favors the conditional objective for the desired control over the output by conditioning. Our transitional training procedure only requires minimal changes in the architecture of the existing state-of-the-art GAN model.

We demonstrate the advantage of the proposed method over the existing ones, by evaluating our method on four benchmark datasets under the limited data setup. The major contributions of this study are summarized as follows:

- We identify and characterize the problem of conditioning-induced mode collapse when training class-conditional GANs under limited data setups.
- We propose a training method for class-conditional GANs that exploits the training stability of unconditional GANs to mitigate the observed conditioning collapse.
- The effectiveness of the proposed method is demonstrated on four benchmark datasets. The method is shown to significantly outperform the state-of-the-art and the compared baselines.

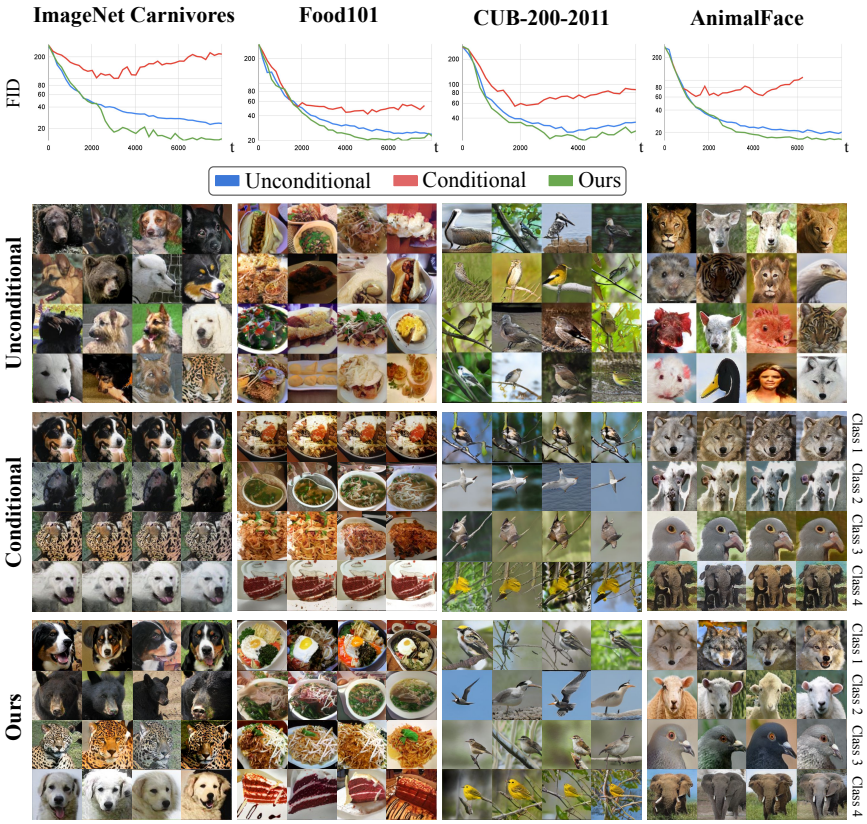
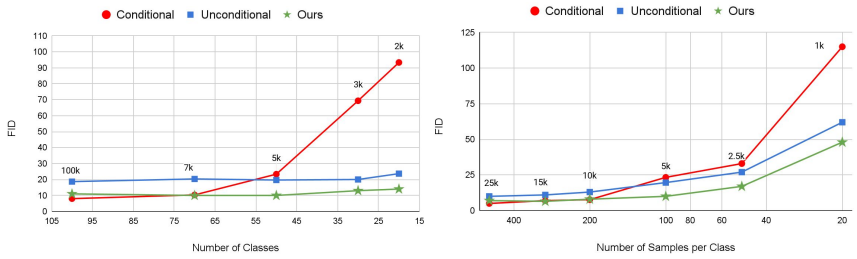


FIGURE 3.1: FID curves (first row) and sample images for training StyleGAN2+ADA unconditionally (second row), conditionally (third row), and using our method (fourth row) on four datasets under the limited-data setup (from left to right: ImageNet Carnivores, Food101, CUB-200-2011, and AnimalFace). The vertical axis of FID plots is in log scale for better visualization.



(a) (b)

FIGURE 3.2: The FID scores for different experiments on ImageNet Carnivores using unconditional, conditional, and our proposed training of StyleGAN2+ADA by varying (a) the number of classes (number of samples per class is fixed at 100) and (b) the number of images per class (the number of classes is fixed at 50). The total number of images for the experiments is shown on the data points. The horizontal axis in (b) is in log scale for better visualization.

3.2 CLASS-CONDITIONING MODE COLLAPSE

Training conditional image generation networks is becoming an increasingly important task. The ability to control the generator is a fundamental feature in many applications. However, even in the context of unconditional GANs, previous studies suggest that class information as extra supervision can be used to improve the generated image quality [97–99]. This, in turn, may set an expectation that the extra supervision by conditioning must not lead to the mode collapse of cGANs in setups where the unconditional GANs succeed. Furthermore, one may also expect to resolve the issue of training cGANs on limited data, to an extent, due to the availability of the additional conditional labels.

As the first part of our study, we investigate the effect of class conditioning on GANs under the limited data setup. We base our experiments on StyleGAN2 with adaptive data augmentation (ADA), which is a recent state-of-the-art method for unconditional and class-conditional image generation under limited-data setup [25][1]. Both unconditional and conditional versions of StyleGAN2 are trained on four benchmark datasets (more details in Sec. 3.4.1), with the setup of this paper. The selected datasets are somewhat fine-grained, where the problem of limited data, concerning their availability and labeling difficulty, is often expected to be encountered.

1 We originally also considered BigGAN [18] as another baseline. However, we found it to struggle with limited data in both unconditional and conditional settings.

In Fig. 3.1, we analyze the training of conditional and unconditional GANs by plotting the Fréchet inception distance (FID) [91] during training. The analysis is performed on four different datasets, each containing between 1170 and 2410 images for training. Contrary to our initial expectation, the conditional version consistently yields worse FID compared to the unconditional one during the training. To analyze the cause, we visualize samples from the best model attained during training, in terms of FID, in Fig. 3.1. For each dataset, the unconditional model learns to generate diverse and realistic images, while lacking the ability to perform class-conditional generation. On the other hand, the conditional model suffers from severe mode collapse. Specifically, the intra-class variations are very small and mainly limited to color changes, while retaining the same structure and pose. Moreover, the images lack realism and contain pronounced artifacts.

Next, we further characterize the mode collapse problem observed in Fig. 3.1 by analyzing its dependence on the size of the training dataset. To this end, we employ the ImageNet Carnivores dataset [100], which includes a larger number of classes and images. We perform the analysis by gradually reducing the size of the training set in two ways. In Fig. 3.2a, we reduce the number of classes while having 100 training images in each class. In Fig. 3.2b, we reduce the number of images per class while using 50 classes in all cases. In both cases, the conditional GAN achieves a better FID for larger datasets, here above 5k images. This observation is in line with previous work [97–99]. However, when reducing the data size, the order is inverted. Instead, the unconditional model achieves consistently better FID, while the conditional model degrades rapidly.

Inspired by these observations, we set out to design a training strategy for class-conditional GANs that eliminates the mode collapse induced by the conditioning. As visualized in Fig. 3.1, our proposed approach, presented next, achieves stable training, leading to low FID. The images generated from our model exhibit natural intra-class variations with substantial changes in pose, appearance, and color. Furthermore, our training strategy outperforms both the conditional and unconditional models in terms of FID in a wide range of dataset sizes, as shown in Fig. 3.2.

3.3 METHOD

3.3.1 *From Unconditional to Conditional GANs*

As the analysis in Sec. 3.2 reveals, class-conditional GAN training leads to mode collapse when training data is limited, while the unconditional counterpart achieves good performance for the same number of training samples. This discovery motivates us to design an approach capable of leveraging the advantages of unconditional

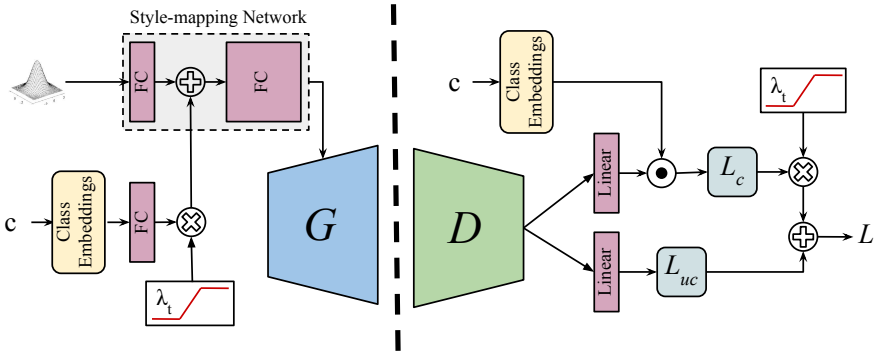


FIGURE 3.3: The proposed modified training objective and architecture of StyleGAN2 allows for transitioning from the unconditional to the conditional model during the training.

training in order to prevent mode-collapse in class-conditional GANs. Our initial inspections of the generated images during conditional training indicate that the mode collapse appears from the very early stages of the training. On the contrary, the corresponding unconditional GAN learns to generate diverse images with gradually improved photo-realism during training. In order to avoid mode-collapse, we aim to exploit unconditional learning at the beginning of the training process. Unconditional training in the early stages allows the GAN model to learn the distribution of the real images without the complications caused by conditioning. We then introduce the conditioning in the later stages of the training. This allows the network to adapt to conditional generation in a more stable way by exploiting the partially learned data distribution.

In general, we control the transition from unconditional training to conditional training using a transition function $\lambda_t \geq 0$. The subscript t denotes the iteration number during training. Specifically, $\lambda_t = 0$ implies a purely unconditional learning, i.e. the conditional information does not affect the generator or discriminator networks. Our goal is to design a training strategy capable of gradually incorporating conditioning during training by increasing the value of λ_t. While any monotonically increasing function λ_t may be chosen, we only consider a simple linear transition from 0 to 1 as,

$$\lambda_t = \min\left(\max\left(\frac{t - T_s}{T_e - T_s}, 0\right), 1\right) . \tag{3.1}$$

Here, T_s and T_e respectively denote the time steps, at which the transition starts and ends. More details on the proposed transition function are provided in appendix B.1. We achieve the desired transition during training by introducing a method for

controlling the behavior of the generator and the discriminator using the function λ_t. An overview of our approach, detailed in the next sections, is illustrated in Fig. 3.3.

3.3.2 *Transition in the Generator*

The generator $G(z)$ of a GAN is trained to map the latent vector z, drawn from a prior distribution $p(z)$, to a generated image $I_g = G(z)$ belonging to the distribution of the training data. A conditional generator $G(z, c)$ additionally receives the conditioning variable c as input, aiming to learn the image distribution of the training data conditioned on c.

The transition from an unconditional $G(z)$ to a conditional $G(z, c)$ generator seemingly requires a discrete architectural change during the training process. We circumvent this by additionally conditioning our generator on the transition function λ_t as $G(z, c, \lambda_t)$. More specifically, we gradually incorporate conditional information during training by using a generator of the form,

$$G(z, c, \lambda_t) = G(S(z) + \lambda_t \cdot E(c)) \,. \tag{3.2}$$

Here, S and E are neural network modules that transform the latent and condition vectors, respectively. In case of $\lambda_t = 0$, the conditional information is masked out, leading to a purely unconditional generator $G(S(z))$. During the transition step $T_s < t < T_e$, the importance of the conditional information is gradually increased during training.

To construct the generator in eq. (3.2), we perform a minimal modification to the original class-conditional StyleGAN2 [32], where the generator consists of a style-mapping network and an image synthesis network. The style-mapping network maps the input latent vector z and the embedding of the condition c to the intermediate representation w, known as style code. The image synthesis network then generates images from the style codes. As illustrated in Fig. 3.3, we modify the conditioning in the generator by feeding the class embeddings to a fully-connected layer, which is then weighted by λ_t before adding to the output of the style-mapping network's first layer.

3.3.3 *Transition in the training objective*

The unconditional and conditional GAN frameworks also differ in their training objectives. The latter's objective assesses the realism of an image based on its conditioning variable. We propose using both unconditional and conditional objectives and follow the same transition as in the generator. Our proposed loss function includes the unconditional objective during the whole training. The conditional term,

on the other hand, is weighted by the transition function λ_t. Our total objective is thus given by,

$$\begin{aligned} L^D &= L^D_{uc} + \lambda_t \cdot L^D_c \,, \\ L^G &= L^G_{uc} + \lambda_t \cdot L^G_c \,. \end{aligned} \tag{3.3}$$

Here, L^D_{uc}, L^D_c, and L^D are the unconditional, conditional, and proposed losses for the discriminator, respectively. Similarly, L^G_{uc}, L^G_c, and L^G represent the unconditional, conditional, and proposed losses for the generator.

As the discriminator is responsible to predict the scores needed for calculating the loss, the proposed training objective also requires modifying the discriminator's architecture. In this regard, the necessary modification must provide both conditional and unconditional scores. We propose using two prediction branches, separately dedicated to conditional and unconditional cases, in the last layer of the discriminator, as shown in Fig. 3.3.

The proposed training objective and discriminator for StyleGAN2 are also visualized in Fig. 3.3. In the discriminator of StyleGAN2, conditioning is performed by matching the features of the input images with the target class embedding. In other words, the conditional discriminator assigns scores to the input images by calculating the dot-product between the features of the input images and the embedding of the target class c. The proposed discriminator architecture bears some resemblance to the architecture of projection discriminators [101]. In contrast to our approach, the projection discriminator aggregates the unconditional and conditional scores inside the discriminator before the loss function. Additionally, the projection discriminator does not perform any transition between the two scores.

3.4 EXPERIMENTS

In this section, we first provide the details of our experimental setup. Then, we present the quantitative and qualitative results of the proposed method, as well as the comparison with existing methods. Finally, we provide more ablation and analysis of different components of our method.

3.4.1 *Experiment Setup*

Datasets: We use four datasets to evaluate our method: ImageNet Carnivores [100], CUB-200-2011 [102], Food101 [103], and AnimalFace [86]. To keep our experiments in the limited-data regime, we decrease the number of classes and images per class in some of these datasets using random sampling.

- *ImageNet Carnivores* is a subset of the ImageNet dataset [65], which contains 149 classes of carnivore animals. The images are further processed to only contain the animal faces. We use a subset of the dataset with 20 classes and 100 images per class.
- *Food101* contains 101 food categories, having a total amount of 101k images. We use a subset of the dataset with 20 classes and 100 images per class.
- *CUB-200-2011* contains 200 categories of different bird species with around 60 images per class. We use a subset of this dataset containing 20 classes with all the images in each class.
- *AnimalFace*, similar to ImageNet Carnivores, is a dataset that contains images of animal faces. However, the animals in AnimalFace are not limited to carnivores. AnimalFace contains 20 classes with 2432 images in total. Since AnimalFace is already a small dataset, we do not further reduce its size.

Implementation details: We base our method on the official PyTorch implementation of StyleGAN2+ADA. The hyper-parameter selection for the base unconditional and conditional StyleGAN2 is performed automatically as provided in the official implementation. Training is done with a batch size of 64 using 4 GPUs. For the transition function, we use $T_s = 2\text{k}$ and $T_e = 4\text{k}$ in all experiments.

Evaluation Metrics: We evaluate our method using Fréchet inception distance (FID), as the most commonly-used metric for measuring the similarity between the distribution of real and generated images. As FID can be biased when real data is small [25], we also include kernel inception distance (KID) [104] as a metric that is unbiased by design.

3.4.2 *Results and Comparisons*

To assess the efficacy of the proposed method, we provide a quantitative comparison with the well-established baselines and existing methods:

- *BigGAN+ADA [18]*: Achieving outstanding results on ImageNet, BigGAN has been widely used for class-conditional image generation (more details in Sec. 3.5). We use the implementation with ADA provided by [105].
- *ContraGAN+ADA [105]*: A class-conditional model based on BigGAN that outperforms BigGAN in many setups using self-supervision in the discriminator (more details in Sec. 3.5).
- *U-StyleGAN+ADA:* Unconditional training of StyleGAN2+ADA using the original unconditional architecture.

- *C-StyleGAN+ADA:* Conditional training of StyleGAN2+ADA using the original conditional architecture.
- *C-StyleGAN+ADA+projD:* The modified version of conditional StyleGAN2 with ADA by replacing its discriminator with the projection discriminator [101].
- *C-StyleGAN+ADA+Lecam:* Regularizing C-StyleGAN2+ADA using Lecam regularizer, recently proposed by [95] for limited-data setup (more details in Sec. 3.5). The authors have suggested a hyper-parameter in the range of $[0.1, 0.5]$. As we did not observe a noticeable difference between different values in the suggested range, we always set the hyper-parameter to 0.3.

Results are reported in Tab. 3.1. BigGAN and ContraGAN, even though coupled with ADA, struggle to achieve any good generation quality in our experiments. Conditional StyleGAN2 shows better results on three of the datasets compared to that of the other two conditional competitors. However, the FID and KID scores are still very high. As discussed before, C-StyleGAN is consistently outperformed by its unconditional counterpart. Replacing StyleGAN2's discriminator with the projection discriminator does not yield a noticeable advantage over the original architecture, as it brings improvements on two of the datasets, but degrades the performance on the other two. Adding Lecam regularization to the C-StyleGAN2 shows promising results, achieving good FID and KID scores on Food101 and AnimalFace. However, it still fails to achieve as good generation quality as the unconditional StyleGAN2. The FID and KID scores for our proposed method indicate a significant and consistent advantage over all the compared methods in all four datasets. Our method is able to maintain a stable training and achieve better generation quality than both unconditional and conditional StyleGAN2.

Method	**Carnivores**		**Food101**		**CUB-200-2011**		**AnimalFace**	
	FID	KID	FID	KID	FID	KID	FID	KID
BigGAN+ADA	97	0.0665	111	0.0794	136	0.0860	90	0.0587
ContraGAN+ADA	97	0.0629	124	0.0961	137	0.0934	89	0.645
UC-StyleGAN2+ADA	23	0.0093	24	0.0071	27	0.0059	20	0.0048
C-StyleGAN2+ADA	100	0.0493	42	0.0135	55	0.0197	61	0.0107
C-StyleGAN2+ADA+ProjD	103	0.0503	32	0.0108	54	0.0182	71	0.0186
C-StyleGAN2+ADA+Lecam	62	0.0211	27	0.0086	37	0.0179	26	0.0042
Ours	**14**	**0.0021**	**20**	**0.0045**	**22**	**0.0032**	**16**	**0.0018**

TABLE 3.1: Comparison of the proposed method with baselines and existing methods on four datasets in terms of FID and KID metrics.

The FID curves during training along with the generated examples using the proposed method are visualized in Fig. 3.1. FID curves indicate training dynamics as stable as the unconditional training while achieving better FID. In addition, the generated images of our method are clearly of more diversity and quality compared to those of the standard conditional model, showing the advantage of the proposed method. The results in Fig. 3.2 further demonstrate a clear advantage of our method over a wide range of data sizes. Our approach maintains the performance of cGANs for larger datasets, while significantly outperforming the unconditional counterpart when data is more scarce. This shows that our method enables cGANs to use the additional label information to achieve better generation quality without falling into the mode collapse induced by conditioning.

3.4.3 *Ablation and Analysis*

In this section, we provide further ablation and analysis over different components of our method. First, we provide an ablation study containing four different variants:

- *No transition:* Training the modified architecture with the new objective, without any transition in the objective or the generator (equivalent to using an auxiliary unconditional loss term to train a conditional model).
- *Transition only in G:* Performing the transition only in the generator (Sec. 3.3.2), while the training objective is the summation of the unconditional and conditional term.
- *Transition only in loss:* Performing the transition only in the training objective (Sec. 3.3.3), while the generator is fully conditional from the beginning.
- *Final method:* The final method with all the proposed components.

Tab. 3.2 presents the results of the ablation study on Food101 and ImageNet Carnivores. The *No transition* version yields poor results, showing that the mode collapse is not alleviated by only adding an auxiliary unconditional training objective. Adding the transition to the generator already brings significant improvement to the model. Having the transition only in the objective, on the other, does not lead to good results. To our initial surprise, this reveals that transitioning in the generator is a crucial part of the method. However, transitioning in the objective in addition to that in the generator, as proposed in our final method, achieves the best results.

Next, we analyze the impact of *when* the transition between unconditional and conditional learning is applied. The total transition time is fixed to $T_e - T_s = 2\text{k}$ time steps. We then report the results for the Food101 and ImageNet Carnivores datasets

	Food101		ImageNet Carniv.	
Experiment	**FID**	**KID**	**FID**	**KID**
No transition	79	0.0300	110	0.0436
Transition only in G	25	0.0064	17	**0.0019**
Transition only in loss	80	0.0297	107	0.0539
Final method	**20**	**0.0045**	**14**	0.0021

TABLE 3.2: Ablation study over different components of the proposed method, including the proposed architecture and objectives, as well as the transition in the generator and in the objective.

	Food101		ImageNet Carnivores	
Experiment	**FID**	**KID**	**FID**	**KID**
$T_s = 0$	24	0.0068	27	0.0075
$T_s = 1k$	22	0.0057	15	0.0023
$T_s = 2$k	20	0.0045	14	0.0021
$T_s = 4$k	21	0.0056	14	0.0021
$T_s = 6k$	23	0.0056	15	0.0028

TABLE 3.3: Analyzing the importance of the transition starting time (T_s). The transition period is constant at 2k for all the experiments.

for different starting times T_s in Tab. 3.3. Importantly, we notice significantly worse results if the transition is started at the beginning of the training $T_s = 0$. This further supports the hypothesis that conditioning leads to mode collapse in the early stages of the training. By introducing conditional information in a later stage, good FID and KID numbers are obtained without being sensitive to the specific choice of T_s. In Tab. 3.4, we further independently analyze the transition end time T_e, while keeping $T_s = 2$k fixed. Again, our approach is not sensitive to its value. Our method, therefore, does not require extensive hyper-parameter tuning.

Lastly, we visualize the evolution of the generated images and the formation of classes during training. Fig. 3.4 shows how images generated from the unconditional phase of training on AnimalFace start to evolve into different images of the class Panda. In addition to the formation of the classes, Fig. 3.4 shows how the image quality continues to improve during and after the transition. In appendix B, more ablation studies (appendices B.6 and B.7), as well as images generated with our method (appendix B.8) are provided for further assessment of the proposed method.

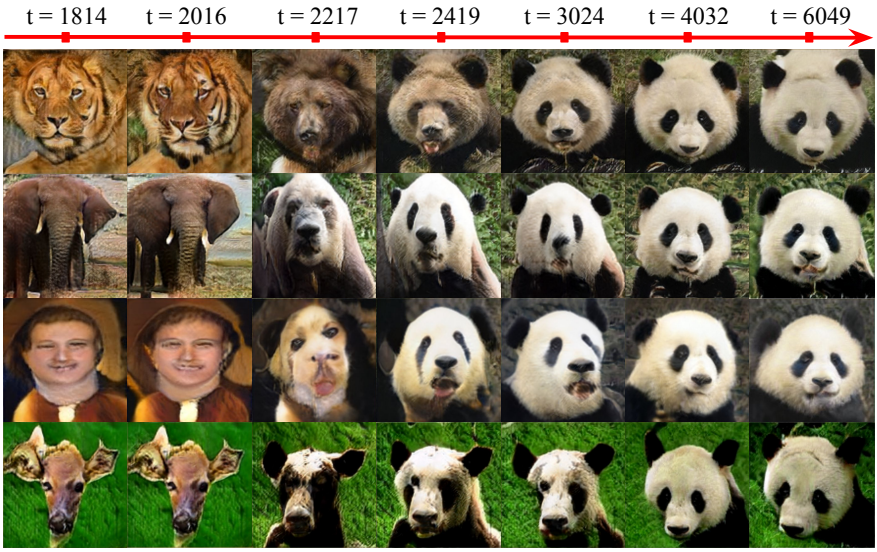


FIGURE 3.4: Visualization of the formation of the class “Panda" in AnimalFace during the transition from unconditional to conditional training. The transition starts at $t = 2$k.

3.5 RELATED WORK

Class-conditional GANs: The first conditional GAN architecture, introduced by [92], incorporated conditioning by concatenating the condition variable to the input of the generator and the discriminator. AC-GAN [106] equipped the discriminator with an auxiliary classification task to ensure the conditional generation. cGAN with projection discriminator [101] proposed a new discriminator architecture, ensuring the class conditioning by computing the dot-product between image features and class embeddings. Improving over cGAN with projection discriminator, BigGAN [18] was able to become the state-of-the-art cGAN on ImageNet dataset [65]. Inspired by the recent progress in self-supervised learning, ContraGAN [105] improved over BigGAN by exploiting an auxiliary self-supervised task in the discriminator for better image representation learning. StyleGAN [19], mainly known for unconditional image generation, was extended to class conditioning in the improved version, and coupled with adaptive differentiable augmentation (ADA), outperformed the state-of-the-art cGANs in the small data setup [25].

cGANs in small data regimes: There exist several directions to address the problem of training GANs on small data. Transfer learning (TL) exploits large pre-

	Food101		ImageNet Carnivores	
Experiment	**FID**	**KID**	**FID**	**KID**
$T_e = 3k$	21	0.0053	16	0.0032
$T_e = 4$k	20	0.0045	14	0.0021
$T_e = 5k$	23	0.0062	15	0.0029

TABLE 3.4: Analyzing the importance of the transition ending time (T_e). The starting time (T_s) is constant at 2k for all the experiments.

training data to provide better initialization for learning the target data. Several works recently have studied the best practices for TL in GANs [60–64]. As an example, cGANTransfer [107] proposed class-specific knowledge transfer in class-conditional GANs by learning the target class embeddings as a linear combination of the ones in the pre-trained model. Although effective, TL usually requires large pre-training data with sufficient domain relevance to the target. Data augmentation (DA) is another technique for addressing small data. To prevent DA from leaking to the generated images, recent works proposed differentiable DA [25, 87]. Adding adaptive differentiable augmentation (ADA) to StyleGAN2 resulted in a significant improvement in conditional generation from small datasets, outperforming previous models on CIFAR10 [25]. In addition to the aforementioned methods, there are other studies focusing on better architecture or objective design for small data regimes. [108] proposed a lighter unconditional architecture and a self-supervised discriminator for StyleGAN. [95] proposed the Lecam regularization to prevent the discriminator from over-fitting on small data by penalizing the current difference between real and fake predictions from previous fake and real predictions, respectively.

Conditioning collapse in GANs: Previous studies have reported a decrease in diversity in tasks with strong pixel-level conditioning, such as semantic masks or images (e.g. super-resolution) [20, 109–111]. Such lack of diversity is generally considered to be due to the conflict between the adversarial and reconstruction losses common in image-conditional GANs. The same effect, however, is not only unexplored for the class-conditional setting but also does not directly translate to this setup. In this work, we discover the conditioning collapse for class-conditional GANs to occur when training data is small (Fig. 3.2).

3.6 CONCLUSIONS

In this chapter, we studied the problem of training class-conditional generative adversarial networks with limited data. Our empirical study demonstrated that class-conditioning can lead the training of GANs to mode collapse within the investigated setup. To prevent such collapse, we presented a method of injecting the class conditioning by transitioning from unconditional to the conditional setting, in an incremental manner. To enable such transition, we have proposed architectural modifications and training objectives, which can be easily adapted by any existing GAN model. The proposed method achieves outstanding results compared to the state-of-the-art methods and established baselines, for the limited data setup across four benchmark datasets. In the future, we will study our method for other types of conditioning (e. g., semantics and images), as well as other architectures.

4

EFFICIENT 3D-AWARE GENERATION WITH CONVOLUTIONS

Pose-conditioned convolutional generative models struggle with high-quality 3D-consistent image generation from single-view datasets, due to their lack of sufficient 3D priors. Recently, the integration of Neural Radiance Fields (NeRFs) and generative models, such as Generative Adversarial Networks (GANs), has transformed 3D-aware generation from single-view images. NeRF-GANs exploit the strong inductive bias of neural 3D representations and volumetric rendering at the cost of higher computational complexity. This chapter aims at revisiting pose-conditioned 2D GANs for efficient 3D-aware generation at inference time by distilling 3D knowledge from pretrained NeRF-GANs. We propose a simple and effective method, based on re-using the well-disentangled latent space of a pre-trained NeRF-GAN in a pose-conditioned convolutional network to directly generate 3D-consistent images corresponding to the underlying 3D representations. Experiments on several datasets demonstrate that the proposed method obtains results comparable with volumetric rendering in terms of quality and 3D consistency while benefiting from the computational advantage of convolutional networks.

4.1 INTRODUCTION

Generative Adversarial Networks (GANs) [15] have made outstanding progress in photorealistic image generation and manipulation in many applications [18, 20, 21, 32, 112–115]. Recently, there has been an increasing interest in extending GANs to 3D-aware generation from single-view image datasets, with the goal of providing disentangled control over the content and the viewpoint of the generated images.

Image GAN models have been historically based on convolutional architectures, enabling efficient training and generation for 2D tasks. However, pose-conditioned convolutional GANs (pcGANs) struggle with 3D-consistent image generation due to their lack of sufficient 3D priors [116]. Therefore, some studies have previously attempted to disentangle the pose from the content in pcGANs using explicit 3D supervision [117–119], which, however, is not readily available for most datasets. As a result, later methods moved away from fully convolutional GANs by incorporating 3D inductive biases in the architecture and training pipeline, such as 3D neural representations and differentiable rendering methods [116, 120–122].

The advent of Neural Radiance Fields (NeRFs) [123] has recently transformed the neural 3D representation and the task of novel-view synthesis [124–130]. For this reason, NeRFs have been successfully integrated with GANs to achieve promising results in 3D-aware generation [131–137]. NeRF-GANs, in their general form, map a latent space to a 3D representation of objects and generate images from queried viewpoints using volumetric rendering. However, volumetric rendering is computationally demanding due to its ray-casting process, making high-resolution generation both slow and memory-intensive. Recent works have proposed different approaches to improve the computational efficiency of NeRF-GANs using more efficient 3D representations [132, 136, 137] and training protocols [132, 133]. Nevertheless, volumetric rendering remains an integral part of these models.

In recent NeRF-GANs [132, 133, 136, 137], convolutional networks have been reintroduced in the generator architecture as super-resolution networks or as 3D-representation generators, in order to scale up NeRF-GANs for high-resolution generation. In this study, we take a different approach to integrating NeRF-GANs and convolutional GANs for 3D-aware generation from single-view image datasets. In particular, we investigate the capacity of convolutional generators to achieve 3D-consistent rendering with explicit pose control when learning from a pre-trained NeRF-GAN without any additional explicit 3D supervision. A convolutional generator that fairly preserves the 3D consistency, image quality, and the correspondence between the generated images and the underlying 3D representation can be used for efficient multi-view inference in setups where volumetric rendering is not affordable, such as in mobile applications. However, balancing and minimizing the trade-off between efficiency and 3D consistency is a highly challenging task, which we set out to explore in this work.

We propose a simple but effective method for distilling a pre-trained NeRF-GAN into a pose-conditioned fully convolutional generator. The main component of our approach is based on exploiting the well-disentangled intermediate latent space of the NeRF-GAN in the convolutional generator. In particular, our convolutional generator learns to map each latent code from the 3D generator, along with the target viewpoint, to the corresponding obtained images by explicit volumetric rendering. By doing so, we aim to distill the NeRF-GAN's underlying 3D knowledge into the convolutional generator, as well as to establish a correspondence between the images of the generator and the 3D representation of the NeRF-GAN. As demonstrated in Fig. 4.1, our experiments on three different datasets indicate that the convolutional generator trained with our method achieves comparable results to volumetric rendering in terms of image quality and 3D-consistency, while benefiting from the superior efficiency of convolutional networks.

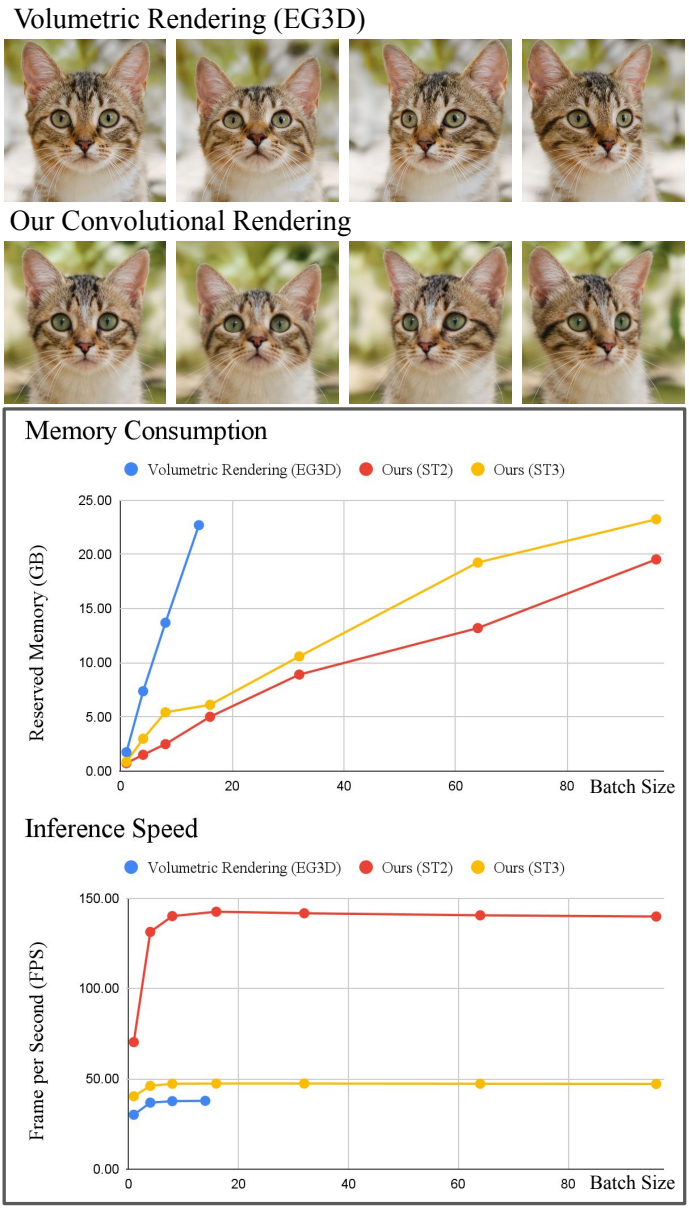


FIGURE 4.1: Top: Views of the same subject cat generated by a volumetric rendering generator (EG3D) and by our convolutional generator. Bottom: Comparison of the inference memory consumption and speed (on a fixed GPU budget) for the two methods.

Our contributions are summarized as follows:

- We propose a method to distill NeRF-GANs into convolutional generators for efficient 3D-aware inference.
- We provide a simple and effective method to condition the convolutional generator on the well-disentangled intermediate latent space of the NeRF-GAN.
- Through experiments on three different datasets, we show that the generator trained by our distillation method significantly preserves the 3D consistency, image quality, and semantics of the pre-trained NeRF-GAN.

4.2 RELATED WORK

3D-Aware Generation from Single-View Images: Prior works have attempted to create 3D awareness in 2D GANs using explicit 3D supervision, such as 3D models [117, 119], pose and landmark annotations [138, 139], and synthetic data [118]. In many applications, obtaining such 3D supervision is not practical. As a result, later works aimed at unsupervised methods by introducing 3D inductive biases in GANs, including 3D neural representations and differentiable rendering [116, 120–122]. These methods, although promising, lag far behind 2D GANs in terms of image quality or struggle with high-resolution generation due to the additional computational complexity.

NeRF-GANs: NeRFs have shown outstanding potential in compactly representing 3D scenes for novel view synthesis. GRAF [131] and Pi-GAN [132] are the first works to integrate NeRFs and GANs. While achieving highly consistent 3D-aware generation, the computational restrictions of the NeRF framework make these methods impractical for high-resolution generation or environments with constrained resources. In order to extend NeRF-GANs to higher resolutions, convolutional super-resolution networks were used in later studies [133, 134, 136], at the expense of some multi-view inconsistencies. EpiGRAF [135], in contrast, adopts an efficient multi-scale patch training protocol, but still requires full high-resolution volumetric rendering for inference, which makes it comparatively more computationally demanding than competitors.

Other studies aim at bringing the recent advances in the efficiency of NeRFs to NeRF-GANs. Although there exist numerous works on efficient 3D representations [126, 140–144] and volumetric sampling [145–149] in NeRFs, only a subset of them [34, 137, 150] have been successfully applied to NeRF-GANs. This is because they are mainly designed for the single-scene setup, making their adaptation to the generative setup not trivial. The use of sparse voxel grids in VoxGRAF [137] and multi-plane image representations in [150] result in efficient and 3D-consistent

generation while compromising the image quality and the 3D geometry. EG3D [34] proposes using tri-planes to represent the geometry of the generated objects. Exploiting tri-planes, coupled with carefully designed techniques to enforce 3D consistency, allows EG3D to significantly improve both computational efficiency and image quality. Live 3D Portrait [151] is a concurrent work based on EG3D that aims at real-time one-shot reconstruction of faces by estimating the canonical tri-planes of a pre-trained EG3D. However, Live 3D Portrait is computationally limited by the underlying volumetric rendering of EG3D. Most similar to our study, SURF-GAN [152] aims to discover directions for pose control in a pre-trained 2D GAN by generating multi-view images using a pre-trained NeRF-GAN. However, using NeRF-GANs only as multi-view supervision does not fully exploit their underlying 3D knowledge. Moreover, the 2D generator obtained using this method does not preserve any correspondence between the NeRF-GAN's 3D representation and the generated images. Different from SURF-GAN, we exploit the intermediate latent space of pre-trained NeRF-GANs to distill 3D knowledge into a 2D generator. This approach establishes a correspondence between the convolutional generator and the NeRF-GAN's 3D representation.

4.3 METHOD

In this section, we first provide a brief overview of the formulation of NeRF-GANs and then explain our proposed formulation in detail.

4.3.1 *Preliminaries*

The general formulation of NeRF-GANs consists of a 3D-representation generator $G^{3D}(z)$, which maps a latent variable z (usually drawn from a normal distribution) to a 3D representation of an object. Then, in order to render an image $I_{z,c}$ from the target viewpoint (camera parameters) $c \in R^{25}$, volumetric rendering is applied to the generated 3D representation. We base our method on EG3D [34], as it provides a strong trade-off in image quality, 3D consistency, and efficiency among recent NeRF-GANs.

EG3D represents 3D scenes using tri-planes, which are three axis-aligned orthogonal feature planes, each with a size of $N \times N \times C$, where N is spatial resolution and C is the number of channels. To represent a 3D position $x \in R^3$, x is projected onto each of the three feature planes, retrieving the corresponding feature vector (F_{xy}, F_{xz}, F_{yz}) via bilinear interpolation, and aggregating the three feature vectors via summation. To obtain the color and density at position x, a lightweight MLP

decodes the feature vector obtained for the queried position to a density and color value.

The tri-plane generator $G^{3D}(z,c)$ in EG3D consists of a mapping network $M^{3D}(z,c)$, which maps the input latent code and the target viewpoint to an intermediate latent variable w, namely the style code. The style code then is used to modulate a convolutional synthesis network $S^{3D}(w)$ to generate the tri-planes $I^{3p}_{z,c}$. In order to render an image $I_{z,c}$ from the target viewpoint c, hierarchical volumetric rendering is applied to the tri-planes. Since volumetric rendering at high resolutions is computationally too expensive, EG3D does so at a lower resolution and uses a convolutional super-resolution network to obtain a final image. More specifically, the low-resolution output of volumetric rendering in EG3D consists of a 32-channel feature map $I^{f}_{z,c}$, the first three of which represent the low-resolution RGB image $I^{LR}_{z,c}$, which is given as input to the super-resolution network. EG3D is trained in an adversarial fashion with a viewpoint-conditioned dual discriminator D that ensures the photorealism of the generated images from the target viewpoints, as well as the consistency between high-resolution and low-resolution images.

4.3.2 *Convolutional Rendering of pre-trained NeRF-GANs*

The aim of the proposed method is to distill a pre-trained NeRF-GAN $G^{3D}_{z,c}$ into a 2D image generator $G^{2D}_{z,c}$, such that $G^{2D}_{z,c}$ directly predicts 3D-consistent multi-view images $I'_{z,c}$, corresponding to the volumetric renderings obtained by the underlying 3D representation of $G^{3D}_{z,c}$. To this end, we propose to exploit the well-disentangled style space of G^{3D} to distill the underlying 3D representation into G^{2D}. Sharing the style space w of the pre-trained 3D generator with the convolutional renderer is the first step towards establishing a correspondence between the 3D representations and the generated images. Secondly, it allows training the convolutional generator for 3D-consistent generation without the need for generating multiple views of the same objects and enforcing multi-view consistency.

The overall architecture of EG3D and our convolutional generator is visualized in Fig. 4.2. The convolutional generator is based on StyleGAN architecture [25, 153], consisting of a mapping network, a low-resolution convolutional feature prediction, and a convolutional super-resolution network. The mapping network transforms the style code w of G^{3D} and the target viewpoint c to the style code w' of G^{2D}. Then, the low-resolution feature predictor S^{2D} estimates the EG3D features and low-resolution image obtained by the volumetric rendering. Estimated features and images are then mapped to the high-resolution outputs using the super-resolution network. In our setup, the super-resolution network is initialized with EG3D's super-resolution network and is jointly optimized with the feature predictor network.

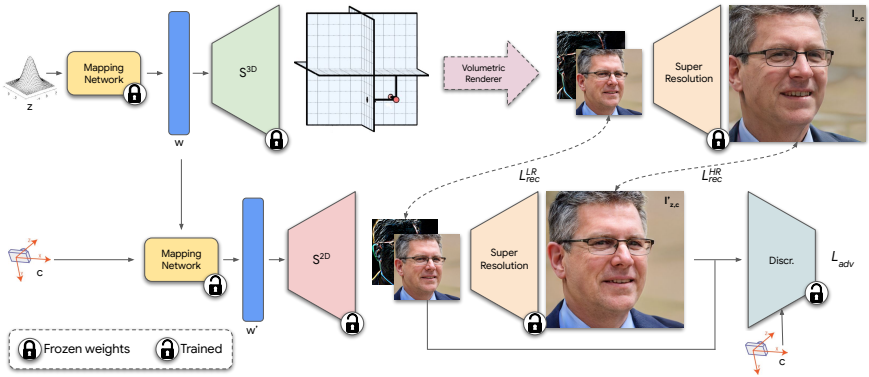


FIGURE 4.2: Using a student-teacher framework, we distill 3D consistency from a frozen volumetric rendering based NeRF-GAN (top) to a 2D convolutional renderer (bottom). A loss consisting of high- and low-resolution image reconstructions and an adversarial component allows us to retain good image quality and 3D consistency.

4.3.3 *Training*

In order to train the proposed convolutional renderer, we use a teacher-student framework, where the volumetric rendering is used to supervise G^{2D} on the viewpoint-conditioned mapping of (z, c) to $I'_{z,c}$. A schematic representation of our training regime is reported in Fig. 4.2. Specifically, for each training sample, we randomly sample z_i and c_i and use the pre-trained NeRF-GAN to obtain: the corresponding style code w_i, the low-resolution image $I^{LR}_{z_i,c_i}$, feature maps $I^{f}_{z_i,c_i}$ rendered by volumetric rendering, as well as the high-resolution image $I^{HR}_{z_i,c_i}$ generated by the super-resolution network. Together, these form a training sample i for the proposed convolutional renderer.

We input z and c into G^{2D} and compute a loss function composed of three parts. We first add a reconstruction term L^{LR}_{rec} between the low-resolution outputs of the volumetric and convolutional renderers. A second reconstruction loss L^{HR}_{rec} is applied between the super-resolved outputs of the two renderers. Lastly, we apply an adversarial term L_{adv}.

In the following, we drop the subscripts z_i, c_i to reduce the clutter in notation. The low-resolution reconstruction loss L^{LR}_{rec} consists of a pixel-wise smooth L1 loss

between the two feature maps, as well as a perceptual loss between the generated and target low-resolution images,

$$\begin{aligned} L_{rec}^{LR} =& \lambda_{L1}^{LR} \cdot \text{Smooth}L1(I'^{f}, I^{f}) + \\ & \lambda_{\text{perc}}^{LR} \cdot \text{PerceptualLoss}(I'^{LR}, I^{LR}). \end{aligned} \tag{4.1}$$

Here, λ_{L1}^{LR} and λ_{perc}^{LR} are the weights for the low-resolution smooth L1 and perceptual loss, respectively.

Similarly, the high-resolution reconstruction loss L_{rec}^{HR} is defined as:

$$\begin{aligned} L_{rec}^{HR} =& \lambda_{L1}^{HR} \cdot \text{Smooth}L1(I'^{f}, I^{f}) + \\ & \lambda_{\text{perc}}^{HR} \cdot \text{PerceptualLoss}(I'^{HR}, I^{HR}). \end{aligned} \tag{4.2}$$

where λ_{L1}^{HR} and λ_{perc}^{HR} are the weights for the high-resolution smooth L1 and perceptual loss, respectively.

The adversarial term L_{adv} is similar to the one used in EG3D. We use the same dual-discriminator architecture D as in EG3D to ensure the realism of the high-resolution images, their consistency with the low-resolution version, and the compliance of the generated image with the queried viewpoints. The total loss L_{total} used for training G^{2D} is,

$$L_{total} = L_{rec}^{LR} + L_{rec}^{HR} + \lambda_{adv} \cdot L_{adv}, \tag{4.3}$$

where λ_{adv} is the weight for the adversarial loss.

Two-Stage Training: In practice, empirical experiments show that training the convolutional renderer using the full objective from the beginning will lead to high-quality but 3D-inconsistent images. Therefore, we instead propose a 2-stage training curriculum. In the first stage, G^{2D} is only optimized by pure distillation of the volumetric rendering G^{3D} using L_{rec}^{LR} and L_{rec}^{HR} until the renderer achieves reasonable generation quality. Then, the adversarial loss L_{adv} is added to the training to further improve the performance. By applying this 2-stage curriculum, we are able to counter the 3D inconsistency induced by the adversarial training.

Pose-Correlated Dataset Bias: As shown in EG3D [34], adversarial training of the convolutional network is prone to learning pose-correlated dataset biases, such as more smiling in non-frontal viewpoints in the FFHQ dataset [154], which in turn results in 3D attribute inconsistencies. To mitigate such biases in the FFHQ dataset, we use both real images and the images rendered from EG3D as the real examples shown to the discriminator. The proportion of the EG3D-rendered images shown to D is controlled by the hyper-parameter α $(0 \leq \alpha \leq 1)$, which is set to 0.5 in our experiments. As we will discuss in Sec. 4.4.6, α can be used to control the trade-off between image quality and 3D consistency.

4.4 EXPERIMENTS

In this section, we first describe our experimental setup for the evaluation of our method. Then, we compare the proposed method with baselines in terms of visual quality, 3D consistency, and computational efficiency. Moreover, we provide an ablation study and a discussion on the benefits and trade-offs of the proposed method.

4.4.1 *Datasets*

Following EG3D [132], we evaluate our method on three datasets:
Flickr-Faces-HQ (FFHQ) [154]: A collection of 70k high-quality images of real-world human faces, as well as corresponding approximate camera extrinsics estimated using an off-the-shelf pose estimator.
AFHQ Cats: A sub-category of the Animal-Face-HQ (AFHQ) [155], consisting of around 5k high-quality images of cat faces, as well as corresponding camera extrinsics estimated using an off-the-shelf pose estimator.
ShapeNet Cars: A category of ShapeNet [156] consisting of synthetic images of cars rendered from different viewpoints, as well as the corresponding camera extrinsics annotations.

4.4.2 *Baselines*

We consider EG3D [34] and the method proposed in SURF-GAN [152] as our main baselines for this study. For a more complete evaluation, we also compare our method to additional relevant baselines:
EG3D [34]: The NeRF-GAN used for distilling 3D knowledge in the convolutional generator. EG3D serves as the upper bound for the 3D consistency of the proposed method.
Pose-Conditioned StyleGAN (PC-GAN): A standard conditional 2D GAN, conditioned on the pose annotations without any knowledge distillation.
SURF: Inspired by the proposed method in SURF-GAN [152], we create a baseline called SURF, where multi-view images of EG3D are used to discover pose-control in a pre-trained 2D StyleGAN.
LiftGAN [121]: A method predating EG3D and SURF baselines based on differentiable rendering that distills a 2D GAN in order to train a 3D generator.

4.4.3 *Implementation and Evaluation Details*

We implement and evaluate the proposed generator using both StyleGAN2 (ST2) [25] and StyleGAN3 (ST3) [153] architectures. For the pre-trained NeRF-GAN, we use the official models from EG3D [34] (for ShapeNet Cars, we re-train the model as the official model does not match the results reported by EG3D). We train our experiments using a batch size of 16. The rendering resolution and the final resolution are (128, 512) for FFHQ and AFHQ and (64, 128) for ShapeNet Cars. Both training and inference experiments were conducted using NVidia RTX 3090 GPUs. In all experiments, we set all of the weights of reconstruction loss terms (λ_{L1}^{LR}, λ_{perc}^{LR}, λ_{L1}^{HR}, λ_{perc}^{HR}) to the value 1 and the weight of the adversarial loss (λ_{adv}) to the value 0.1.

4.4.4 *Metrics*

We evaluate our method quantitatively in terms of visual quality and 3D consistency.
Fréchet Inception Distance (FID) [91]: The most common metric to assess the quality and diversity of generation.
Kernel Inception Distance (KID) [91]: An unbiased alternative to FID for smaller datasets.
Pose Accuracy (PA): Following previous works [34], we measure the ability of the model in generating images of the query poses by calculating the Mean Squared Error (MSE) between the query poses and the pose of the generated images, estimated using an off-the-shelf pose estimator [157].
Identity Preservation (ID): As a metric for 3D consistency, we measure the degree of face identity preservation between different viewpoints with respect to the canonical pose using ArcFace [158] cosine similarity for the FFHQ setup.
3D Landmark Consistency: As another 3D consistency metric, we measure the change in facial landmarks between different viewpoints in FFHQ using MSE. The 3D landmarks are estimated using an off-the-shelf estimator [157].

4.4.5 *Quantitative Comparison*

In the following, we quantitatively compare the proposed method with the baselines described in Sec. 4.4.2 in terms of inference efficiency, visual quality, and 3D consistency.

Method	FFHQ		AFHQ		ShapeNET Cars	
	FID ↓	KID ↓	FID ↓	KID ↓	FID ↓	KID ↓
EG3D [34]	5.0	0.0018	2.9	0.0003	3.5	0.0017
PC-GAN	19.3	0.0085	4.5	0.0009	6.1	0.0018
LiftGAN [121]	29.8*	-	-	-	-	-
SURF	31.1	0.0153	-	-	-	-
Ours (ST2)	6.6	0.0019	3.8	0.0011	3.1	0.0013
Ours (ST3)	6.8	0.0023	3.2	0.0007	3.1	0.0012

TABLE 4.1: Comparison of image quality on three datasets in terms of FID and KID metrics. *The value is borrowed from [121].

4.4.5.1 *Efficiency*

The efficiency of fully-convolutional networks compared to the rendering-based methods is well-known. To better assess the practical benefit of the proposed method, we provide a comparison of inference efficiency between EG3D. Fig. 4.1 visualizes an example of the inference memory consumption and speed of the two methods using different batch sizes on a fixed GPU budget (in this case, on RTX 3090 GPU with 24G of memory). As shown, EG3D is restricted to small batch sizes (a maximum of 14) due to its costly memory consumption, whereas our method can scale up to a maximum of 96 samples per batch ($\sim 7\times$). As for the speed, our generator achieves a better frame-per-second, especially when using StyleGAN2 as its backbone ($> 3\times$).

4.4.5.2 *Image Quality*

To assess the trade-off brought about by our convolutional generator, we evaluate the quality of the generated images. Tab. 4.1 shows the FID and KID scores for our method and the baselines on different datasets. Compared to the PC-GAN and SURF baselines, our method constantly achieves higher quality. This confirms that exploiting the style space of the pre-trained NeRF-GAN contributes to the ability of the convolutional renderer in pose-conditioned generation. Although our method does not fully match the visual quality of EG3D, it is still able to fairly maintain high image quality and significantly reduce the compromise in the quality compared to the other convolutional counterparts.

Method	Pose Acc. ↓	3D Landmark ↓	ID ↑
EG3D [34]	0.002	0.018	0.75
PC-GAN	0.009	0.062	0.56
SURF	0.044	0.014	0.86
Ours (ST2)	0.002	0.023	0.75
Ours (ST3)	0.002	0.022	0.75

TABLE 4.2: Comparison of 3D consistency metrics on FFHQ.

4.4.5.3 *3D Consistency*

While, unlike volumetric rendering, 2D convolutions do not guarantee 3D consistency, we show that our approach achieves a good performance in this regard. We assess the 3D consistency of generated images on FFHQ by measuring the pose accuracy, 3D landmark consistency, and face identity preservation, as discussed in Sec. 4.4.4. Based on the results, which are provided in Tab. 4.2, our method achieves comparable 3D consistency with EG3D, while PC-GAN and SURF struggle. Note that the high values for identity preservation and 3D landmark consistency in SURF are due to the limited pose variations, and hence generating similar images regardless of the input pose, as reflected by the pose accuracy (and the visual examples in Fig. 4.3).

4.4.6 *Ablation*

Ablation on Loss Functions: The proposed training objective in Sec. 4.3.3 consists of different loss terms to ensure both image quality and consistency with the output of volumetric rendering. In this section, we ablate the importance of these components. Tab. 4.3 shows the FID scores for the following experiments on the loss terms on AFHQ dataset:

- **LR**: Low-resolution reconstruction loss with frozen super-resolution network.
- **HR**: Only the high-resolution reconstruction loss.
- **LR + HR** Both low-resolution and high-resolution reconstruction losses.
- **HR + ADV**: Reconstruction and adversarial losses on high-resolution images.
- **Full (LR + HR + ADV):** Full training objective, including the reconstruction and adversarial terms.

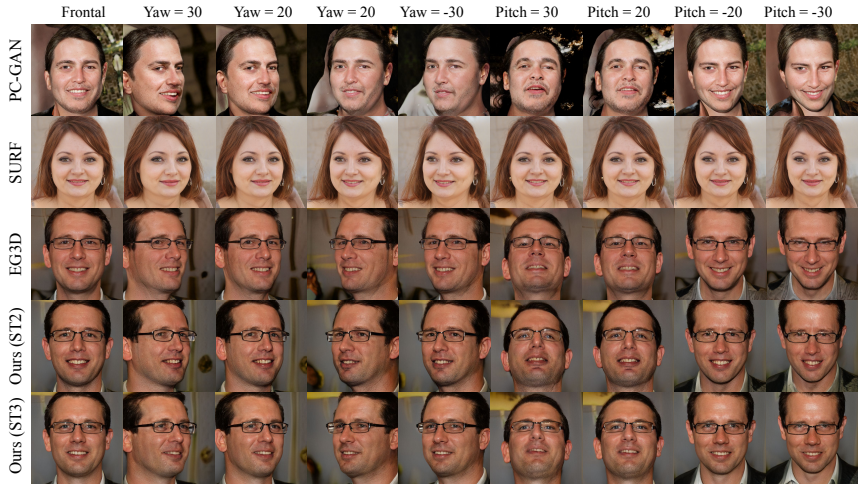


FIGURE 4.3: Qualitative examples of variations in yaw and pitch for FFHQ. Compared to the pose-conditioned GAN and SURF baseline, our proposed method nearly matches the 3D consistency and image quality of volumetric rendering (EG3D).

FIGURE 4.4: Qualitative examples of yaw and pitch variations for AFHQ cats. In line with our quantitative experiments, the pose-conditioned convolutional baseline (PC-GAN) fails to preserve the identity of the subject under different poses. In contrast, our method exhibits similar preservation of identity to the volume rendering approach (EG3D), despite the difference in computational resources and time.

Method	FID ↓
LR	30.55
HR	10.39
LR + HR	9.1
HR + ADV	6.58
Full (LR + HR + ADV)	3.2

TABLE 4.3: Ablation study on different loss functions for training the ST3 convolutional renderer on AFHQ Cats dataset.

Method	Ours (ST2)		Ours (ST3)	
	FID ↓	3D Landmark ↓	FID ↓	3D Landmark ↓
$\alpha = 0$	5.5	0.027	5.7	0.027
$\alpha = 0.3$	6.3	0.023	6.1	0.024
$\alpha = 0.5$	6.8	0.022	6.6	0.023

TABLE 4.4: The effect of mixing real images and EG3D-rendered images as real examples for adversarial training, controlled by the parameter α, on FFHQ dataset.

As shown by the ablation study, the combination of all the proposed loss terms leads to the best FID scores.

Single-Stage Vs. Two-Stage Training: As mentioned in Sec. 4.3.3, we find that single-stage training by jointly optimizing for both reconstruction and adversarial losses results in subtle inconsistencies such as color shifts and geometry warps, which can be mitigated using the proposed 2-stage training in Sec. 4.3. As the observed inconsistencies are difficult to capture using our quantitative 3D consistency metrics, we provide a visual comparison between the examples of single-stage and two-stage training on AFHQ in Fig. 4.5.

Mitigating Pose-Attribute Correlation: In Tab. 4.4, we provide an ablation on the parameter α introduced in Sec. 4.3.3 for FFHQ dataset. As shown, including EG3D-generated images ($\alpha > 0$) improves the 3D consistency at the cost of a lower generation quality.

4.4.7 *Qualitative Comparison*

In this section, we provide a visual comparison of our method with the baselines. In Figs. 4.3 and 4.4, we provide visual examples of variations in yaw and pitch for

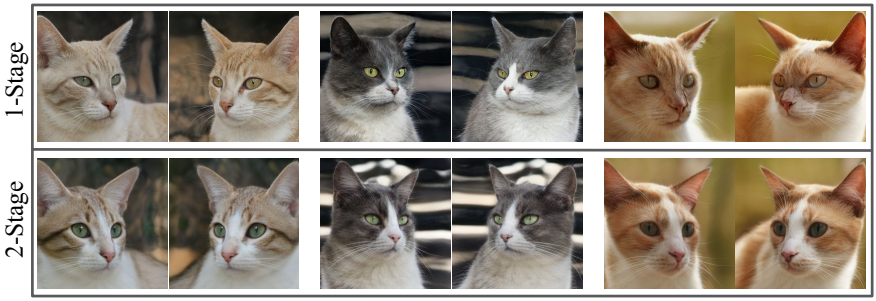


FIGURE 4.5: One-stage training causes subtle color and geometry inconsistencies (first row). Such inconsistencies can be resolved using our proposed 2-stage training (second row).

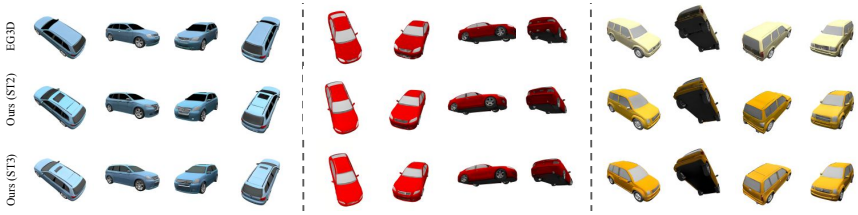


FIGURE 4.6: Qualitative examples of different camera poses in ShapeNet Cars for three different car models.

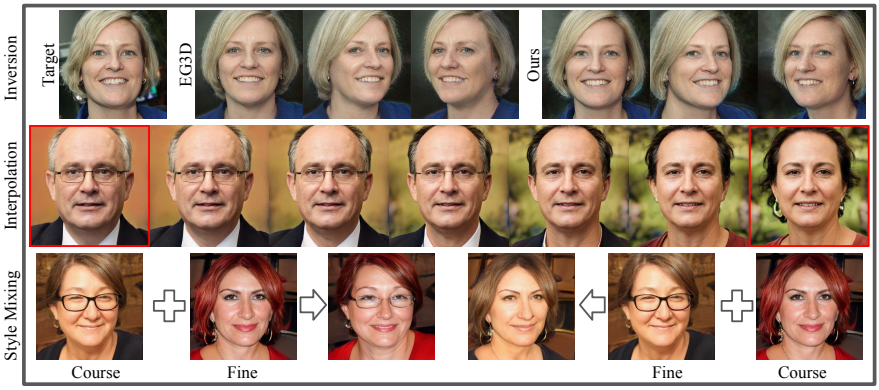


FIGURE 4.7: First row: Inversion using PTI [159] for EG3D and our method; second row: interpolation in the latent space of our method; third row: style mixing in the latent space of our method.

FFHQ and AFHQ Cats. Compared to PC-GAN and SURF, our proposed method closely matches the 3D consistency and maintains the image quality of volumetric rendering. Fig. 4.6 additionally provides examples of ShapeNet Cars generated using our method and their corresponding images from EG3D. Similarly, our method exhibits preservation of 3D consistency and image quality, despite the difference in required computational resources.

4.4.8 *Inversion, Interpolation, and Style Mixing*

As the proposed generator follows a StyleGAN architecture, it can easily benefit from most of the editing techniques common in the GANs' literature. Fig. 4.7 shows examples of inversion using Pivotal Tuning Inversion (PTI) [159], latent space interpolation, and style mixing.

4.4.9 *Discussion: StyleGAN2 Vs. StyleGAN3*

StyleGAN2 is more computationally efficient than StyleGAN3. Based on the provided quantitative evaluations, our method reaches comparable image quality and 3D consistency with both architectures. However, StyleGAN2 is known to suffer from more texture stitching and artifacts [153], which we also observe in the generated images.

4.4.10 *Correspondence between Convolutional and Volumetric Rendering*

As mentioned before, exploiting the style space of the pre-trained NeRF-GAN also provides an opportunity for establishing a direct correspondence between the 3D representation of the 3D generator and the generated images using the convolutional generator. A close comparison of images generated using the convolutional and volumetric rendering in Figs. 4.3, 4.4 and 4.6 indicates that the convolutional render is able to infer and match many attributes of the underlying 3D representation from the shared latent space and generate images similar in content to those of volumetric rendering. However, there still remains a gap in the full correspondence of the two rendering methods, as semantic and identity changes are visible between the corresponding images generated by the two methods. Investigating more explicit approaches for enforcing correspondence could be an interesting direction for improving the convolutional rendering for NeRF-GAN models.

4.5 CONCLUSION

In this chapter, we presented a method to distill a pre-trained NeRF-GAN into a pose-conditioned convolutional generator. The proposed method enables considerably higher efficiency, which is crucial if 3D neural rendering is to become ubiquitous and deployed at scale. To do so, we proposed exploiting the intermediate latent space of the pre-trained NeRF-GAN as a conditioning input for the convolutional generator. We additionally provided a training protocol to further improve the visual quality and 3D consistency of the images generated using our generator. Through our experiments, we showed that our method maintains good image quality and 3D consistency, significantly better than previous fully-convolutional methods, and approaching those of the baseline NeRF-GAN with volumetric rendering. Finally, while our method takes steps toward achieving full correspondence between the two rendering methods, there remains a gap in terms of image semantics. Improving this aspect remains a subject for further research.

5

GENERATIVE 3D OBJECT INSERTION

Recently, methods for 3D scene editing have been profoundly transformed, owing to the use of strong priors of text-to-image diffusion models in 3D generative modeling. Existing methods are mostly effective in editing 3D scenes via style and appearance changes or removing existing objects. Generating new objects, however, remains a challenge for such methods. In this chapter, we introduce InseRF, a novel method for generative object insertion in the NeRF reconstructions of 3D scenes. Based on a user-provided textual description and a 2D bounding box in a reference viewpoint, InseRF generates new objects in 3D scenes. Specifically, we propose grounding the 3D object insertion to a 2D object insertion in a reference view of the scene. The 2D edit is then lifted to 3D using a single-view object reconstruction method. The reconstructed object is then inserted into the scene, guided by priors of monocular depth estimation methods. We evaluate our method on various 3D scenes and provide an in-depth analysis of the proposed components. Our experiments with generative insertion of objects in several 3D scenes indicate the effectiveness of our method compared to the existing methods. InseRF is capable of controllable, 3D-consistent object insertion without requiring explicit 3D information as input.

5.1 INTRODUCTION

Recent advances in novel view synthesis [123] and generative modeling [17, 26] have significantly advanced 3D generation and manipulation methods. This has enabled the development of powerful 3D generative models for applications such as text-to-3D [42, 160, 161], single-image-to-3D [43, 162–165], and 3D editing [44]. In 3D scene editing in particular, remarkable promise has been shown in modifying the appearance of real-world scene representations based on textual and spatial guidance. Nevertheless, methods able to directly generate and edit 3D assets are limited to simple and object-centric scenes [43, 166–168]. For more complex scenes, most recent editing methods rely on editing different views of the scenes using 2D models. A prime example of this is Instruct-NeRF2NeRF [44] (I-N2N), which iteratively edits the input images on which a 3D scene representation is trained with a global textual editing prompt. Although it achieves great results, I-N2N is limited to editing the style and appearance of scenes. When prompted with localized edits or geometry manipulations (such as object removal or insertion), I-N2N often fails

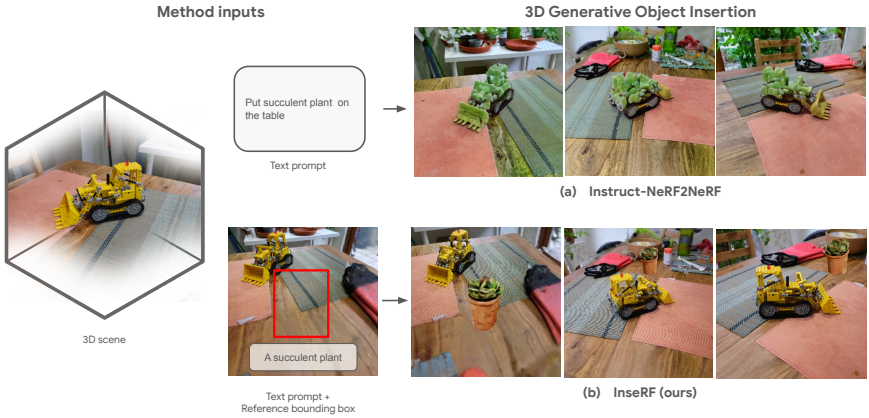


FIGURE 5.1: **Generative 3D object insertion**: while Instruct-NeRF2NeRF allows for altering the overall style of scenes or performing edits with strong positional priors, it fails at inserting objects in arbitrary locations due to the multi-view inconsistency of the edits. Our method, in contrast, is multi-view consistent by design and inserts new objects in user-specified locations.

to perform the desired edits. This is mainly due to the 3D inconsistency of 2D edits across viewpoints and lack of spatial control.

Recent works have aimed at 3D-consistent [45, 46] and localized editing [46, 169, 170] of 3D scenes. Several works have specifically tackled object removal and inpainting in 3D scene representations [47, 48, 171–173]. However, generating and inserting new objects in scenes in a 3D-consistent way remains an open problem and is mainly limited to cases where edits are strongly constrained by spatial priors (e.g. putting a hat on a head or a mustache on a face). Therefore, in this work, we focus on generating and inserting 3D-consistent objects in arbitrary locations in the scene.

Generative object insertion in 3D scenes using 2D generative models is a particularly challenging task, as it requires 3D-consistent generation and placement of objects in different views. A simplistic approach is to separately generate the desired objects using 3D shape generation models [42, 161] and insert them into the scene using 3D spatial information. However, such an approach requires the accurate location, orientation, and scale of the object in 3D, a non-trivial requirement, especially when in contact with other objects in the scene. Moreover, scene-independent generation of the objects can lead to a mismatch between the style and appearance of the scene and the inserted objects. As visualized in Fig. 5.1, we propose a method capable of scene-aware generation of objects in 3D scenes using the textual description of the objects and a single-view 2D bounding box as spatial guidance.

To circumvent multi-view inconsistencies and the need for explicit 3D spatial information, we propose anchoring the 3D insertion by a 2D view of the target object, inserted in one reference view of the scene. Given a 3D reconstruction of the scene, we first render a reference view. Then, conditioned on a text prompt and a 2D bounding box, we use an image editing method to add the target object in the reference view. The generated object is then lifted to 3D using a single-view-to-3D object reconstruction method [43, 162–165]. To place the object in 3D, we propose using the estimated depth of the object in the reference view. After inserting the object in the scene, we apply an optional refinement to the fused scene and objects.

We evaluate our method on several 3D scenes. Our experiments indicate the proposed method's ability to insert diverse objects in 3D scenes, without the need for explicit 3D spatial guidance. To summarize our contributions:

- We address the task of consistent generative object insertion in 3D scenes based on a textual description and a single-view 2D bounding box, which is beyond the capability of the existing 3D scene editing methods.
- We propose a novel method, based on grounding the insertion using a reference 2D edit, that is capable of 3D-consistent object insertion without requiring explicit information for the 3D placement.
- Through experiments and visualizations, we show the advantage of our method for generative object insertion compared to the existing baselines.

5.2 RELATED WORK

Language-based 3D scene editing: 3D scene editing has recently undergone a considerable transformation by incorporating the strong priors of 2D text-conditioned diffusion models into 3D generative modeling [44–46, 169, 170, 174, 175]. Instruct-NeRF2NeRF (I-N2N) [44] proposes an iterative method for 3D scene editing, where different viewpoints of the scene are edited using a text-based 2D editing model and used to fine-tune the scene's NeRF representation. Although highly effective in modifying the existing content, I-N2N often struggles with 3D-consistent and localized edits, especially when instructed to remove objects or create new ones in the scene [45, 48]. To address the view consistency of edits, ViCA-NeRF [45] proposes a method based on viewpoint-correspondence regularization and a strategy to align the latent space of edited and unedited viewpoints. DreamEditor [46] tackles the 3D consistency by adapting the 2D diffusion model to the multi-view images of the scene using DreamBooth [176], as well as identifying a 3D region of interest of an existing object based on text-image semantic similarity. The method in [169] addresses localized editing differently by obtaining a 3D relevance field for the edits

based on the discrepancy between the predictions of the diffusion model with and without instruction conditioning. These methods, despite the improvements, remain limited in their ability to generate new objects, often struggling with cases where a strong spatial prior for the placement of objects does not exist.

Object removal and replacement: Another direction recently explored in the area of 3D scene editing is 3D-consistent removal or replacement of objects in the scenes. Some studies assume having multi-view masks of the target object [47, 171]. For instance, Reference-Guided NeRF Inpainting [47] is a method mainly designed and evaluated for removing objects from forward-facing scenes based on accurate multi-view masks as input. The authors additionally showcase a few examples of replacing existing objects with new ones using their method. Other studies rely on user-provided single-view annotations or text-conditioned segmentation models to automatically obtain multi-view masks of objects to be removed [48, 172, 173, 177]. ReplaceAnything3D [177] is a recent study that enables the replacement of existing objects within a scene. However, such approaches for extracting multi-view masks do not transfer to the task of object insertion, as they rely on the assumption that the objects already exist in the scene.

Generative object insertion: In contrast to global scene editing, object removal, and object replacement, generating objects in 3D scenes is not well-explored in existing works. FocalDreamer [178] is a recent work proposed for adding editable parts to a base 3D mesh. Provided with a text prompt and the rough 3D placement of the target edits, FocalDreamer applies score distillation [42] to add the desired parts to the base shape. Although achieving compelling results, FocalDreamer requires user-provided 3D regions (rotation, translation, and scale), and its generalization beyond base shapes to complex 3D scenes is not investigated. Language-Driven Object Fusion [179] is another recent work that aims at fusing an existing or generated foreground object with a background 3D scene. The authors first adopt a 2D diffusion model for view synthesis from the scene and the object using DreamBooth [176]. Then, conditioned on a user-provided 3D bounding box, the authors propose a pose-conditioned dataset update strategy for the training of the scene NeRF containing the object. The proposed fusion strategy requires users to provide an exact 3D bounding box. In contrast to the existing language-driven object insertion methods, our approach works well with both forward-facing and 360 scenes, and it only requires a rough 2D bounding box from one rendered view of the scene, making it more suitable for real-world applications.

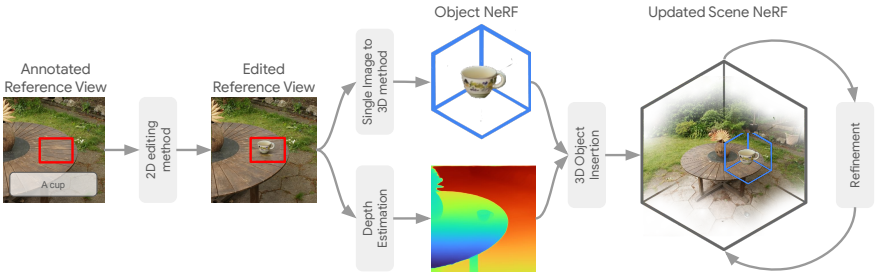


FIGURE 5.2: **Overview of the proposed method.** Given a single reference view, a 2D bounding box, and a text prompt describing the object to be inserted, a 2D edit is generated portraying a view of the object. This 2D edit is then transformed into a 3D model of the object and placed into the scene using the procedure described in Sec. 5.3.4. After the 3D placement, the object and the scene representations are fused as described in Sec. 5.3.5. Finally, an optional refinement can be performed to further improve the appearance.

5.3 METHOD

Our method takes a NeRF reconstruction of a 3D scene, a textual description of the object to be inserted, and a 2D bounding box in a rendered reference view of the scene. As output, it returns a NeRF reconstruction of the same scene containing the generated target 3D object placed in a location guided by the 2D bounding box. It is noteworthy that our method only requires a rough bounding box, as we rely on the priors of diffusion models for the exact 2D positioning.

Our method consists of five steps: 1) creating a 2D view of the target object in a chosen reference view of the scene based on a text prompt and a 2D bounding box; 2) reconstructing a 3D object NeRF from the 2D view previously created; 3) estimating the 3D placement of the object in the scene using monocular depth estimation; 4) fusing the object and scene NeRFs into a single scene containing the object in the estimated placement; 5) applying an optional refinement step to the fused 3D representation to improve the insertion further. Fig. 5.2 shows an overview of our pipeline. Next, we discuss each step in detail.

5.3.1 *Preliminaries*

Diffusion Models: Diffusion models are a type of generative model that maps Gaussian noise to highly realistic and diverse samples. They consist of (1) a forward

process that maps data samples $\boldsymbol{x}_0$ to noise $\boldsymbol{x}_T$, and (2) a backward process that creates data samples from noise.

The forward process consists of T steps $t \in [0, T-1]$:

$$q(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_t \mid \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t \boldsymbol{I}), \tag{5.1}$$

with variances β_t chosen such that the noise $\boldsymbol{x}_T \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$.

The backward process, which is used to generate data samples from Gaussian noise and optionally an additional conditioning signal, has the following shape:

$$q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1} \mid \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t, \boldsymbol{c}), \sum_\theta(\boldsymbol{x}_t, t, \boldsymbol{c})), \tag{5.2}$$

where the parameters of the backward/denoising distributions are predicted by a U-Net, whose weights θ are optimized by increasing the likelihood of the data samples. Diffusion models can be conditioned on different types of signals, such as images or text, as well as masks, and can be extended for different tasks, such as 2D editing [180–182] and inpainting [183, 184].

Neural Radiance Fields: NeRFs are a novel view synthesis method trained on a set of posed images by minimizing the photometric loss between ground truth and rendered pixels. In NeRFs, the density σ and color $\boldsymbol{c}$ at 3D points in space are predicted by a neural function $\boldsymbol{f}_\phi$. The pixel color corresponding to a ray $\boldsymbol{r} = (\boldsymbol{o}, \boldsymbol{d})$ with origin $\boldsymbol{o}$ and viewing direction $\boldsymbol{d}$ can be composed through volumetric rendering. To do so, a set of points along the ray $\boldsymbol{r}_i = \boldsymbol{o} + t\boldsymbol{d}$ is sampled, splitting the ray into a set of intervals $\delta_i = (t_i, t_{i+1}]$. The pixel color of the ray can then be composed as:

$$C(\boldsymbol{r}) \approx \sum_{i=1}^{N} w_i \boldsymbol{c_i}, \tag{5.3}$$

$$w_i = T_i(1 - \exp\{(-\sigma_i \delta_i\})), \tag{5.4}$$

$$T_i = \exp\left\{(-\sum_{j=1}^{i-1} \sigma_j \delta_j)\right\}. \tag{5.5}$$

In the above equations $(\sigma_i, \boldsymbol{c}_i) = \boldsymbol{f}(\gamma(t_i); \phi)$, where γ is the positional encoding function.

5.3.2 *Editing the Reference View*

Our editing pipeline starts by choosing a rendered view of the scene and inserting a 2D view of the target object from a user-provided text prompt and a 2D bounding box. This reference view is then used to anchor the 3D insertion by providing the

desired appearance and location. Since the spatial guidance from text is not sufficient for localized 2D object insertions [182, 185, 186], we specify the object location with a 2D bounding box. To ensure insertion within the bounding box, we opt for a mask-conditioned inpainting method as our 2D generative model. We choose Imagen [187], a powerful text-to-image diffusion model, and further adapt it to mask-conditioning by using RePaint [183].

5.3.3 *Single-View Object Reconstruction*

After obtaining the reference edit, we create a 3D reconstruction of the 2D view of the object generated within the bounding box using the recent paradigm of single-view object reconstruction with 3D-aware diffusion models [43, 162–165]. Such reconstruction methods are typically trained on large-scale 3D shape datasets [188] and therefore contain strong priors over the geometry and appearance of 3D objects. We use SyncDreamer [163] for object reconstruction, as it offers a good trade-off between quality and efficiency.

5.3.4 *3D Placement*

Depth Estimation: The reference 2D bounding box constrains the 3D location of the object to a frustum in the scene. To determine the location of the object in the 3D frustum, we propose using the prior from monocular depth estimation methods. We apply MiDaS [189] on the edited reference image to estimate the depth of the object with respect to the reference camera. As MiDaS provides non-metric depth measurements, we perform an extra depth alignment between the estimated depth of the edited reference view and the reference depth rendered from the scene NeRF by estimating a global scale and shift between the reference and estimated depth maps. Specifically, for a more accurate alignment around the object area, we estimate the alignment parameters using weighted least-square estimation, where measurements are inversely weighted based on their distance to the center of the object bounding box. After the alignment, we use the depth of the center pixel d in the object bounding box as a rough estimate of the object's center in the frustum, which will be further optimized in the next step. More details are provided in appendix D.

Scale and Distance Optimization: Using the estimated depth d as the distance of the object's center from the reference camera helps with resolving the scale-depth ambiguity of the 3D object, but is not accurate enough to closely match the original edit. Additionally, single-view reconstruction methods like SyncDreamer (discussed in Sec. 5.3.3) are trained to generate multi-view images from fixed camera distance r' and focal length f'. In general, as these parameters are different from those of

the reference camera, the reconstructed object NeRF appears with a different scale in the reference view once placed at the estimated distance. Therefore, we propose an additional optimization step for the scale and the distance of the object with two constraints: 1) the object must reside at the estimated depth; 2) the rendered view of the object in the reference camera should match the initial edit in scale and appearance. To ensure a proper initial state for the optimization, we initialize our scale s and object's distance r as:

$$s_0 = \frac{d}{r'} \cdot \frac{f'}{f}, \tag{5.6}$$

$$r_0 = s_0 \cdot l + d, \tag{5.7}$$

where s_0 and r_0 are the initial object scale and distance, and l is the distance of the 3D point corresponding to the center of the bounding box from the origin of the object NeRF's coordinate system. Given a 3D point P' in the original object NeRF's coordinate system, the corresponding 3D point P in the scaled coordinate system is obtained as $P = sP'$.

To obtain the optimized scale s^* and distance r^*, we minimize the Mean Squared Error between the ground-truth 2D edit I_G and the image I_R rendered with the new parameters:

$$r^*, s^* = \arg\min_{r,s} ||I_G - I_R||^2. \tag{5.8}$$

Fig. 5.6 in our ablation study shows the effect of our optimization.

Rotation and Translation: After obtaining the scale and distance of the object from the reference camera, we estimate the placement of the object in the scene by estimating its 3D rotation and translation with respect to the camera. The origin of the object in the scene's coordinate system is obtained as the point along the ray from the reference camera center passing through the center of the bounding box at the desired distance. To obtain the 3D rotation, we align the x-axis of the object's coordinate system to the vector pointing to the reference camera center from the object's origin. More details are provided in appendix D.

5.3.5 *Scene and Object Fusion*

Once the location and orientation of the 3D object in the scene are known, we fuse the NeRF representations of the object and the scene to be able to render views of the scene containing the object. Given a viewpoint, we transform the rays to the coordinate systems of the scene and the object. Each NeRF representation is applied to their transformed rays to predict the color and density of the object and the scene at each 3D point. To render a view using the predictions of the NeRFs, we follow

the strategy of [190], where the density σ_i and color $\boldsymbol{c}_i$ at each 3D point i across a ray in the fused representation are defined as:

$$\sigma_i = \sigma_i^s + \sigma_i^o, \tag{5.9}$$

$$\boldsymbol{c}_i = \frac{\sigma_i^s \boldsymbol{c}_i^s + \sigma_i^o \boldsymbol{c}_i^o}{\sigma_i^s + \sigma_i^o}, \tag{5.10}$$

where σ_i^s and $\boldsymbol{c}_i^s$ are the density and the color of the corresponding sample in the scene NeRF, and σ_i^o and $\boldsymbol{c}_i^o$ those in the object NeRF. To be able to use this formulation in our method for merging the object and the scene, it is crucial to take the scaling of the object's coordinate system into account. Going back to the approximation of the volumetric rendering integration, discussed in Sec. 5.3.1, in eq. (5.4), $\sigma_i \delta_i$ can be seen as the Riemann approximation of the area under the density curve across the ray at interval δ_i. Simply replacing σ_i in eq. (5.4) with the definition in eq. (5.9) results in an inaccurate estimation of the area under the density curve for the merged representation, as the intervals between every two consecutive samples across the rays are not equal between scene and object coordinate systems due to the scaling of the object coordinate system (discussed in Sec. 5.3.4):

$$\delta_i^s = s^* \cdot \delta_i^o, \tag{5.11}$$

where δ_i^s and δ_i^o are the intervals in the scene and object NeRFs, respectively, and s^* is the optimized scale obtained in Sec. 5.3.4. To compensate for the scaling of the intervals, we modify eqs. (5.9) and (5.10) as:

$$\sigma_i = \sigma_i^s + \frac{\sigma_i^o}{s^*}, \tag{5.12}$$

$$\boldsymbol{c}_i = \frac{\sigma_i^s \boldsymbol{c}_i^s + \sigma_i^o \boldsymbol{c}_i^o / s^*}{\sigma_i^s + \sigma_i^o / s^*}. \tag{5.13}$$

As we also show in Fig. 5.7 of our ablation study, this modification is necessary for correct rendering of the fused NeRF.

5.3.6 *Refinement*

As an optional final step, we refine the fused scene and object to improve the imperfections introduced in the initial reference edit and single-view reconstruction. To do so, we adapt the iterative refinement proposed in I-N2N [44] to our setup. First, a set of images is rendered from different views of the fused NeRF. Then the sampled views are further refined using the 2D diffusion model and added to the optimization of the NeRF consecutively. An important difference between our

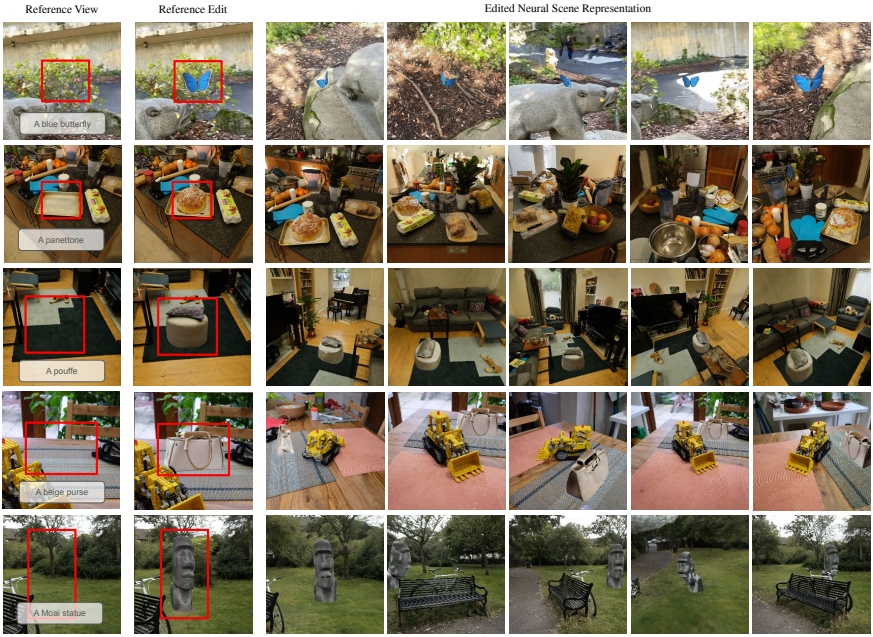
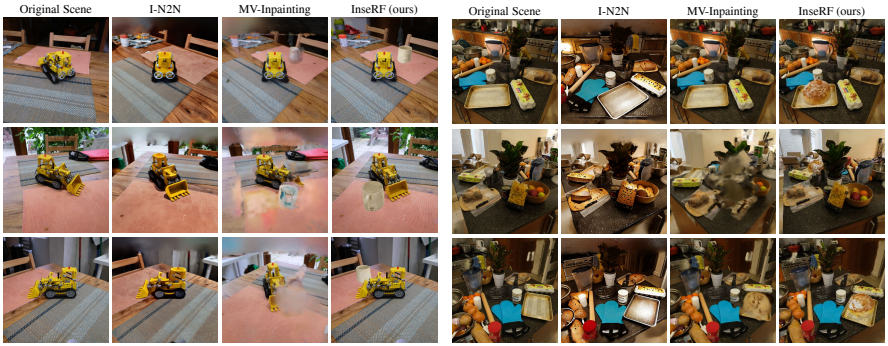


FIGURE 5.3: Examples of using InseRF to insert an object into the neural representation of different indoor and outdoor scenes.

refinement and I-N2N's is that we restrict the refinement to the object region using the object's multi-view masks obtained from the object NeRF for free. Additionally, we adjust our camera trajectory to revolve around the object. We arrange the sampled viewpoints so that the more frontal views are prioritized during NeRF optimization. We find such adjustments to enhance our refinement. The effect of refinement is shown in Fig. 5.8 of our ablation. More details are provided in appendix D.

5.4 EXPERIMENTS

In this section, we explain our training and evaluation procedures. Moreover, we provide the results of our evaluation and comparison with baselines. Finally, we provide an ablation study and analysis of the components of our method.



(a) A mug on the table (b) A panettone on the tray

FIGURE 5.4: **Qualitative comparison** of object insertion with baselines. I-N2N modifies existing objects instead of inserting new ones; MV-Inpainting fails to create geometry at desired locations; Our method, in contrast, can insert new 3D-consistent objects at desired locations.

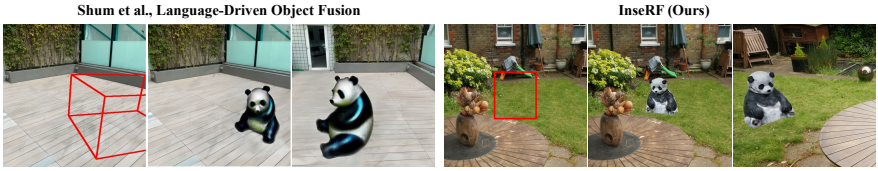


FIGURE 5.5: Comparison with **Language-Driven Object Fusion** [179], which requires a 3D bounding box for spatial guidance: *a sitting panda sculpture*.

5.4.1 *Experimental Details*

Implementation Details: For the NeRF representation of objects and scenes, we use MipNeRF-360 [191] adapted to the hash grids introduced in Instant-NGP [127]. For a more exhaustive description of the implementation of our method, we kindly refer the reader to appendix D.

Datasets: We evaluate our method on a subset of real indoor and outdoor scenes from datasets proposed in MipNeRF-360 [191] and Instruct-NeRF2NeRF [44].

Baselines: We compare our proposed method with:

- **Instruct-NeRF2NeRF (I-N2N)** [44]: We choose I-N2N as our main baseline, as it is a recent and well-established method for 3D scene editing.
- **Multi-View Inpainting (MV-Inpainting)**: We propose another baseline that follows the refinement strategy in I-N2N, but is additionally provided with

accurate multi-view masks for the target object. It is worth emphasizing that, in contrast, our method only requires a rough 2D bounding box in a single reference view.

- **Language-Driven Object Fusion** [179]: We additionally provide a preliminary comparison with a recent work, Language-Driven Object Fusion [179], which requires a 3D bounding box as input spatial guidance. We compare with one of their provided examples by applying the corresponding edit prompt to one of our scenes.

More details on baselines are provided in appendix D.

5.4.2 *Visual Results and Comparisons*

Visual Examples: To assess the ability of our method in generative object insertion, we provide visual examples of applying our method to different 3D scenes in Fig. 5.3. As shown, our method can insert 3D-consistent objects in the scenes. Notably, our method is capable of inserting objects on different surfaces, which is a challenging task in the absence of precise 3D placement information.

Visual Comparison: In Fig. 5.4, we provide a visual comparison with the baselines discussed in Sec. 5.4.1. Attempting to insert new objects in the scene using I-N2N often results in global changes in the scene and modifying existing objects toward the target instead of creating new ones (e. g. I-N2N changes the Lego truck toward a mug in Fig. 5.4a and the items on the kitchen counter toward a panettone in Fig. 5.4b). Using multi-view masks in the MV-Inpainting baseline helps with limiting the 2D edits to the object region and provides strong spatial guidance. However, 2D edits remain inconsistent from different viewpoints. Therefore, using the edits to optimize the NeRF representation results in 3D floaters and failure to generate the target object in a 3D-consistent way. Fig. 5.5 additionally shows a preliminary comparison the recent method, Language-Driven Object Fusion [179], which requires a 3D bounding box as input spatial guidance. In contrast, our method can perform localized modification in the scene and insert 3D-consistent objects using only one single-view bounding box as spatial guidance. More visual results and comparisons are provided in appendix D.

5.4.3 *Quantitative Comparison*

We also provide a quantitative evaluation of the proposed method and its comparison with our baselines. We follow a similar evaluation protocol as I-N2N [44]. We evaluate the methods using three different metrics:

- **CLIP Text-Image Similarity (Text-Image)**: The cosine similarity between the CLIP [192] embeddings of the edit prompt (e.g. "A blue cup") and the images rendered from different viewpoints of the edited scene.
- **Directional Text-Image Similarity (Directional)**: Given a text description of the original scene (e.g. "A table") and an edit prompt describing the edited scene (e.g. "A table with a mug on top"), it measures the similarity of the direction of change from the original to the edited scene between the image and text CLIP embeddings.
- **Temporal Direction Consistency (Temporal)**: Given two adjacent rendered viewpoints of original and edited scenes, this metric measures how much the change of image embeddings between the two viewpoints in the edited scene is consistent with the one in the original scene.

We provide the results of our quantitative evaluation on 15 different edits (5 different scenes) in Tab. 5.1. All metrics are based on cosine similarity, which ranges from -1 to 1. We bring the values between 0 and 1 (the higher the better) for ease of comparison. As depicted, our method effectively outperforms the baselines in all evaluated metrics.

5.4.4 *Ablation and Analysis*

Scale and radius optimization: In Fig. 5.6, we provide a visual ablation demonstrating the importance of the scale and radius optimization proposed in Sec. 5.3.4, where we compare the object placement in the scene using the initial estimations (r_0 and s_0) with the placement after applying the extra optimization. As shown, the initial estimation would only result in a rough and inaccurate placement of the object. With the proposed optimization, our method can insert objects with the scale and depth matching those of the reference view.

Method	Text-Image ↑	Directional ↑	Temporal ↑
I-N2N [44]	0.61	0.53	0.60
MV-Inpainting	0.62	0.50	0.75
InseRF (ours)	**0.63**	**0.57**	**0.81**

TABLE 5.1: **Quantitative comparison** of InseRF and its baselines on different metrics following a similar evaluation protocol as I-N2N [44]. We perform the evaluation on 15 different edits (5 different scenes). Our method outperforms the baselines in all metrics.

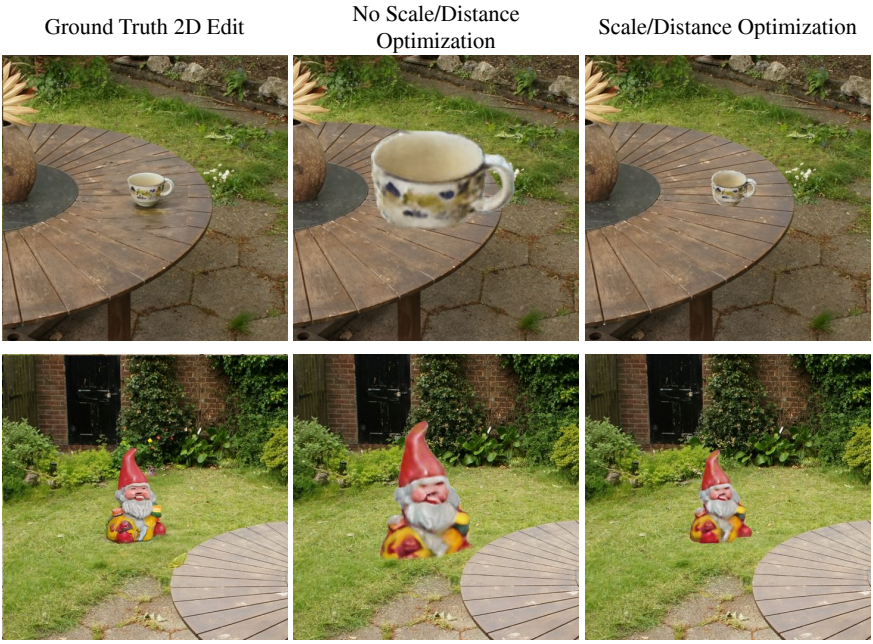


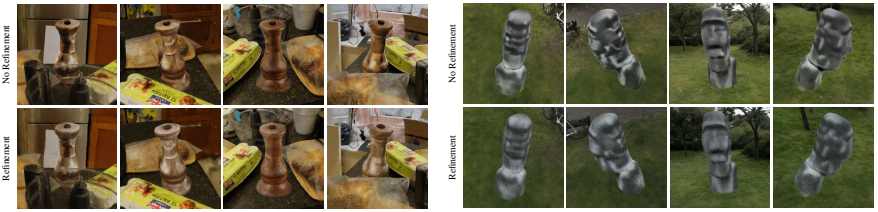
FIGURE 5.6: The effect of **scale and distance optimization**. The object placement is more realistic and faithful to the original edit when the optimization is performed to improve the alignment.

Object density scaling: In Sec. 5.3.5, we proposed a strategy to fuse the NeRF representations of the scene and the object that takes the scaling of the object into account. In Fig. 5.7, we visualize the importance of our adapted formulation for accurate rendering of the inserted objects.

Refinement: In Sec. 5.3.6, we proposed an optional refinement step after inserting the objects in the scenes. Fig. 5.8 shows examples of the effect of the refinement. As shown, the additional refinement can improve some of the details of the inserted objects, such as the lighting and the texture.

Inserting multiple objects: Fig. 5.9 shows our method can be readily used to insert multiple objects in a scene.

FIGURE 5.7: The effect of **density scaling** on fusing object and scene. When the object NeRF scale is not accounted for in the volumetric rendering, the object is not properly displayed in the rendered views.



(a) A pepper grinder (b) A Moai statue

FIGURE 5.8: Visualization of the effect of the **refinement** step. Our refinement can add additional texture details and lighting effects.

5.4.5 *Limitations and Future Work*

Underlying generative models: Our method is a general pipeline for generative object insertion built on top of existing 2D and 3D generative models, which can be easily replaced. The performance of our method is limited by the underlying genera-

FIGURE 5.9: **Inserting multiple objects:** a teapot and a sugar bowl.

tive models, such as the 2D diffusion model or the single-view object reconstruction method. On the other hand, given our general formulation, future improvements in such models readily transfer to our pipeline.

Object lighting: As shown in Fig. 5.8, the refinement step from Sec. 5.3.6 further improves the realism of the insertions. However, our method currently does not explicitly adjust the lighting of the inserted objects to match the lighting conditions of the 3D scenes. As explicitly estimating the ambient light in such complex scenes only from input images (or their NeRF reconstructions) is not trivial, exploring potential ways of further harmonizing the generated objects with the underlying 3D scenes would be a promising future direction for improving the realism of the insertions. In this study, we primarily focus on the orthogonal direction of 3D-consistent object generation in 3D scenes without 3D spatial input, and our method can serve as a foundation for future studies.

5.5 CONCLUSION

We introduced InseRF, a method for generative object insertion in 3D scenes. InseRF takes as input a textual description of the desired object and a 2D bounding box in a single reference viewpoint of the scene. Based on these inputs, InseRF generates a 3D-consistent object in the scene. To do so, InseRF leverages the priors of 2D diffusion models and single-view object reconstruction methods. The proposed method includes several steps necessary to integrate these methods for the task of in-scene object generation. Through evaluations and visualizations on different 3D scenes, we showed the ability of InseRF to generate 3D-consistent objects in the scene without requiring explicit 3D placement information.

ACKNOWLEDGMENTS OF CONTRIBUTIONS

The work presented in this chapter is part of a collaborative project with equally significant contributions from Mohamad Shahbazi (ETH Zurich) and Liesbeth Claessens (ETH Zurich). Mohamad Shahbazi's independent contributions include defining the problem, conducting the literature review, designing the overall method, and implementing the initial version of the pipeline. The joint contributions of Mohamad Shahbazi and Liesbeth Claessens include addressing and improving the initial method's shortcomings, conducting experiments, performing the ablation study and visualizations, and co-authoring the publication submission.

6

CONCLUSION

In this thesis, we addressed different aspects of 2D and 3D generative models under real-world considerations, including controllability, as well as data and computational efficiency. In particular, we focused on different conditional generative models, including class-conditioned, pose-conditioned, and text-conditioned ones, as we were interested in controllable generation. Here, we summarize the contributions of the thesis and provide a discussion on the potential future directions for research on generative models.

SUMMARY OF CONTRIBUTIONS

In Chapter 2, we studied the problem of conditional GAN transfer by transferring the knowledge across both source and target classes. We represented the knowledge of individual classes by their respective batch normalization parameters, which are used for conditioning during the generation. To propagate the knowledge to new classes, we introduced a method, called cGANTransfer, that learns to update and combine the batch normalization parameters of the source classes. The evaluations on three standard benchmarks demonstrate a clear advantage of our method, both in terms of training efficiency and the image generation quality (measured by FID and KMMD), compared to the state-of-the-art methods. Our ablation study showed the importance of jointly using the update and combination steps, which we referred to as sharing and propagation, respectively.

In Chapter 3, we studied the problem of training class-conditional generative adversarial networks with limited data. Our empirical study demonstrated that class-conditioning can lead the training of GANs to mode-collapse within the investigated setup. To prevent such collapse, we presented transitional-cGAN, a method of gradually injecting the class conditioning by transitioning from unconditional training to the conditional case. To enable such a transition, we proposed architectural modifications and training objectives, which can be easily adapted by any existing GAN model. The proposed method achieves outstanding results compared to the state-of-the-art methods and established baselines, for the limited data setup of four benchmark datasets.

In Chapter 4, we presented a method for distilling a pretrained NeRF-GAN into a pose-conditioned convolutional generator. The proposed method enables

considerably higher efficiency, which is crucial if 3D neural rendering is to become ubiquitous and deployed at scale. To do so, we proposed exploiting the intermediate latent space of the pretrained NeRF-GAN as a conditioning input of the convolutional generator. We additionally provided a training protocol to further improve the visual quality and 3D consistency of the images generated using our generator. Through our experiments, we showed that our method maintains good image quality and 3D consistency, significantly better than previous existing fully-convolutional methods and approaching the performance of the baseline NeRF-GAN.

Finally, we introduced InseRF in Chapter 5. InseRF is a method specifically designed for generative object insertion in 3D scenes. InseRF takes as input a textual description of the desired object and a 2D bounding box in a single reference viewpoint of the scene. Based on the provided inputs, InseRF generates a 3D-consistent object in the 3D scene. To do so, InseRF relies on the priors of 2D diffusion models and single-view object reconstruction methods. The proposed method includes various steps necessary to integrate such methods for the task of in-scene object generation. Through evaluations and visualizations on various 3D scenes, we showed the ability of InseRF in the 3D-consistent generation of objects in scenes without requiring explicit 3D placement information.

OUTLOOK AND FUTURE DIRECTIONS

Data-Efficient Generation

Since the introduction of cGANTransfer and transitional-cGAN, generative models have progressed significantly in their capabilities in adapting to small custom data. Recently, many methods have been proposed to exploit the strong prior knowledge of large pretrained text-to-image diffusion models for adaptation to smaller datasets. In particular, a new direction addresses the customization of such models to new concepts (*e.g.*, new objects and identities) using only a few examples, enabling diverse generation of the new concepts in new contexts [176, 193, 194]. Additionally, recent methods such as ControlNet [195] and LoRA [196] provide parameter-efficient strategies for fine-tuning diffusion models on custom data without losing the generalizability of the underlying foundation model. Further exploration of such methods, especially in more extreme cases such as one-shot customization while preserving the attributes of the reference concept in the novel generated samples, would be an interesting future direction.

Computational Efficiency

As discussed in Chapter 1, diffusion models, as the current state-of-the-art generative models, still suffer from slow inference due to their iterative process. Recent methods, such as SnapFusion [197] and MobileDiffusion [198] have proposed lightweight architectures and faster sampling techniques for more efficient inference of diffusion models. In the 3D domain, Gaussian Splatting [199] has recently emerged as an efficient framework for representing 3D scenes. Gaussian Splatting could potentially bring significant improvements to the computational efficiency of 3D generative models. However, its adaptation to the generative setup is not trivial. Despite the preliminary attempts [199, 200], there still exist many opportunities for improving the 3D generative models based on Gaussian Splatting.

Temporal Generation

Finally, generative models have been extensively explored for various 2D and 3D applications. Exploring the temporal dimension of generative models is another important research direction. Recently, several diffusion-based methods for video generation and editing have been proposed [201–205]. However, these methods are mainly limited to generating short sequences or editing only object-centric videos while still struggling with temporal consistency. To go one step further, developing generative models for dynamic 3D scenes remains fairly unexplored. Existing methods for 4D generation and editing are mainly limited to dynamic asset generation [206] or possess significantly restricted editing capabilities [207]. Consequently, exploring 4D generative models could open the door to a variety of new applications for generative models.

A

APPENDIX: CLASS-SPECIFIC TRANSFER LEARNING IN GANS

This appendix includes the additional material for chapter 2. First, we provide more details on the model architecture and implementation of our experimental setup. Then, we further discuss the quantitative results of CIFAR100 [89] experiments. Moreover, we discuss source-to-target similarities, training curves, and single-class target data. Finally, we provide more visual results obtained using our method.

A.1 ADDITIONAL IMPLEMENTATION DETAILS

Architecture: We follow [62, 64] to employ the architecture of BigGAN [18] as our backbone for the GAN transfer tasks. It is worth mentioning that the BigGAN implementation for CIFAR uses the basic form of BigGAN, which for example, does not use a hierarchical latent variable. In particular, Tabs. A.1 and A.2 show the network architecture for CIFAR100 and ImageNet [65] setups. Fig. A.1 shows the architecture of the residual blocks used in the generator and the discriminator. The detailed diagram for the conditional batch normalization layer with knowledge propagation across classes has been provided in Fig. 2.4.
Baselines: The original study proposing batch normalization adaptation (BSA) [64] uses supervised loss function (L1/Perceptual loss) instead of adversarial loss. In our experiments, whenever we refer to BSA, we mean training the GAN model adversarially, while freezing the filters and learning the BN parameters from scratch. Therefore our implementation of BSA could be considered as the main baseline for our experiments since it shares the same setup as ours, but without performing any knowledge transfer across the classes.
Ablation Study Experiments: Tab. 2.2 shows the results for the ablation study on the ImageNet-to-Places365 setup. In the table, “Prior" refers to the BN parameters of the previous classes. The table includes the results for using the prior with and without being further updated via knowledge sharing using target classes. “Shared W" in the fourth experiment refers to using shared similarity weights over all layers of the generator to combine previous BN parameters of each layer. The term “w/o reg" in the fifth experiment refers to not using l1 regularization on the combination weights and l2 regularization on the residuals.

$z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$
Linear $4 \times 4 \times 256$
ResBlock up 256
ResBlock up 256
ResBlock up 256
BN, ReLU, Conv 3×3, Tanh

$x \in \mathbb{R}^{32 \times 32 \times 3}$
ResBlock down 64
ResBlock down 128
ResBlock down 256
ResBlock down 512
ResBlock 1024
ReLU, Global Sum Pooling
Embed(y).h + (Linear$\rightarrow$1)

TABLE A.1: The network architecture for CIFAR setup: Left: the generator. Right: the discriminator.

$z \in \mathbb{R}^{120} \sim \mathcal{N}(0, I)$
Linear $4 \times 4 \times 256$
ResBlock up 256
ResBlock up 256
ResBlock up 256
BN, ReLU, Conv 3×3, Tanh

$x \in \mathbb{R}^{128 \times 128 \times 3}$
ResBlock down 96
None-Local Block (64×64)
ResBlock down 192
ResBlock down 384
ResBlock down 768
ResBlock down 1536
ResBlock 1536
ReLU, Global Sum Pooling
Embed(y).h + (Linear$\rightarrow$1)

TABLE A.2: The network architecture for ImageNet setup: Left: the generator. Right: the discriminator.

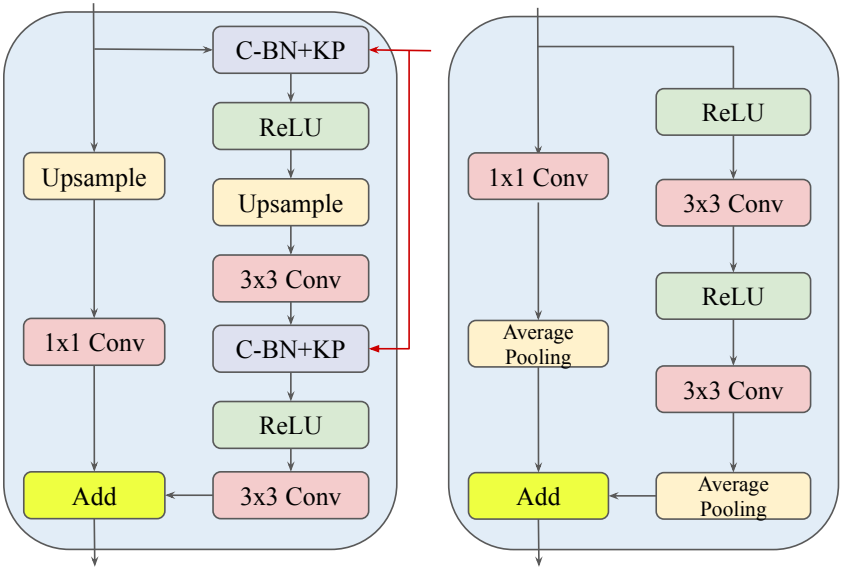


FIGURE A.1: The architecture of the residual blocks used in the network. Left: the generator's ResBlock ("C-BN+KP" layer indicates the conditional batch normalization with knowledge propagation across classes. Please refer to Fig. 2.4 for more details). Right: the discriminator's ResBlock.

A.2 FURTHER DISCUSSION ON QUANTITATIVE RESULTS

In Tab. 2.1, which shows the FID scores for different experiments on CIFAR100, the results of the first experiment (20 classes, 600 samples per class) are marginally different from those of the other experiments. It can be seen that, learning from scratch performs better than all of the transfer learning methods, since the training data is large enough for learning the filters from scratch. However, by reducing the sample number in the next experiments, the performance of learning from scratch immediately deteriorates, while the transfer learning methods remain more robust. However, after reducing the training data even more (20/100, 10/600, 10/300, 10/100), fine-tuning (TransferGAN [60]) also degrades significantly compared to BSA and our method, leading to mode collapse. Comparing the FID scores of BSA and our method on CIFAR, although comparable, it can be observed that our method starts to perform better in the experiments with smaller amounts of data, showing the importance of using prior knowledge from previous classes when training data is small.

FIGURE A.2: Top 3 contributing classes (planetarium, Bird House, Mountain Tent) from the pre-trained network toward the target class “Arch" in Places365 dataset [90]. The classes are selected based on the learned similarity scores of the first layer. Each row depicts one class, and the images are generated from the network pre-trained on ImageNet.

FIGURE A.3: Top 3 contributing classes (Buckeye, Football Helmet, Impala) from the pre-trained network toward the target class “Deer" in Animal Face dataset. The classes are selected based on the learned similarity scores of the first layer. Each row depicts one class, and the images are generated from the network pre-trained on ImageNet.

A.3 DISCUSSION ON CLASS SIMILARITIES

Our method proposes knowledge transfer from previous classes by learning similarity scores over previous BN parameters and combining the BN parameters using those scores to construct the BN parameters of the new classes. Previous classes can contribute to a target class in terms of semantics, shape, texture, or color in a hierarchical manner from bottom to top layers. As an example, Fig. A.2 shows the top 3 ImageNet classes of the pre-trained network (planetarium, Bird House, Mountain Tent) contributing to the target class “Arch" in Places365, based on the similarity weights learned for the first layer (the first layer is generally more interpretable in terms of class similarities, since it is responsible for determining the general structure of the output images, as shown in Fig. 2.9). As it can be seen, these classes contain visual features close to arch structures that can meaningfully be used to generate images from the target class. Fig. A.3 shows another example on Animal Face dataset [86] by visualizing the top 3 classes (Buckeye, Football Helmet, Impala) contributing to the target class “Deer". In this example, we can see that the third class “Impala" is semantically very close to the target class. However, the contribution of the first two classes is not as clear as the previous example. These classes might be contributing to the background, or this might be due to the fact that the similarity scores are actually learned to combine the pseudo-classes, and this does not always guarantee semantic similarity to the initial pre-training classes.

A.4 FID AND LOSS CURVES

As an example of how the losses and the FID scores evolve during the training, the curves for one of the CIFAR100 experiments (Exp. 10/600) have been provided in Fig. A.4. The convergence speed-up is clearly depicted in the FID curve, whereas the same is difficult to derive from the loss plots (due to the adversarial training).

A.5 SINGLE-CLASS TARGET

Although the main focus of our work is multi-class to multi-class knowledge transfer using knowledge propagation and knowledge sharing, the proposed method is not only limited to the multi-class target. As an example, the results of knowledge transfer to the single class “Arch" in Places365 are provided in Tab. A.3.

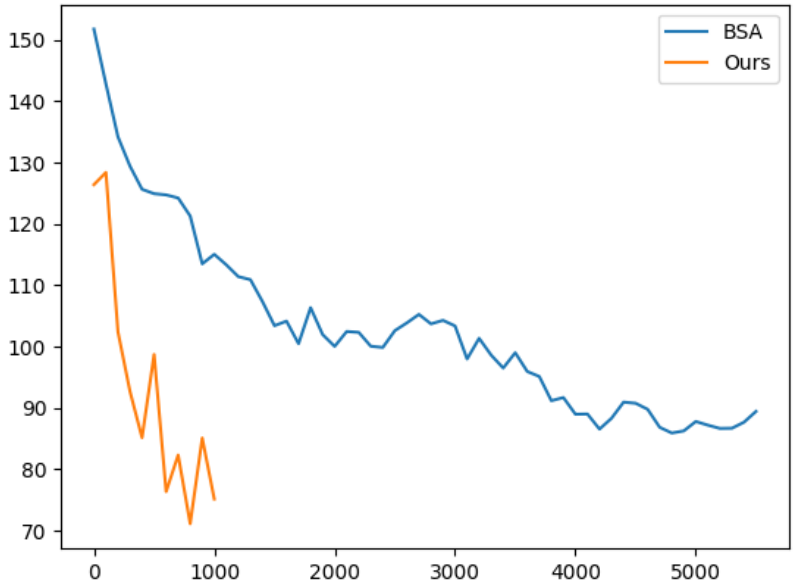


FIGURE A.4: The FID curve (left), the G loss (middle), and the D loss (right) for our method and BSA on CIFAR(10/600).

Method	FID	Iterations
BSA	104	4300
Ours	**78**	**500**

TABLE A.3: FID scores and number of iterations for knowledge transfer from ImageNet to the single target class “Arch" in Places365.

A.6 ADDITIONAL VISUAL RESULTS

Fig. A.5 and Fig. A.6 show additional visual results obtained from BSA (no knowledge propagation across classes) and our method on ImageNet setup. Regarding CIFAR setup, we visualize the results of the experiments 20/300 and 10/300 for BSA and our method in Figs. A.7 and A.8, as examples of CIFAR experiments.

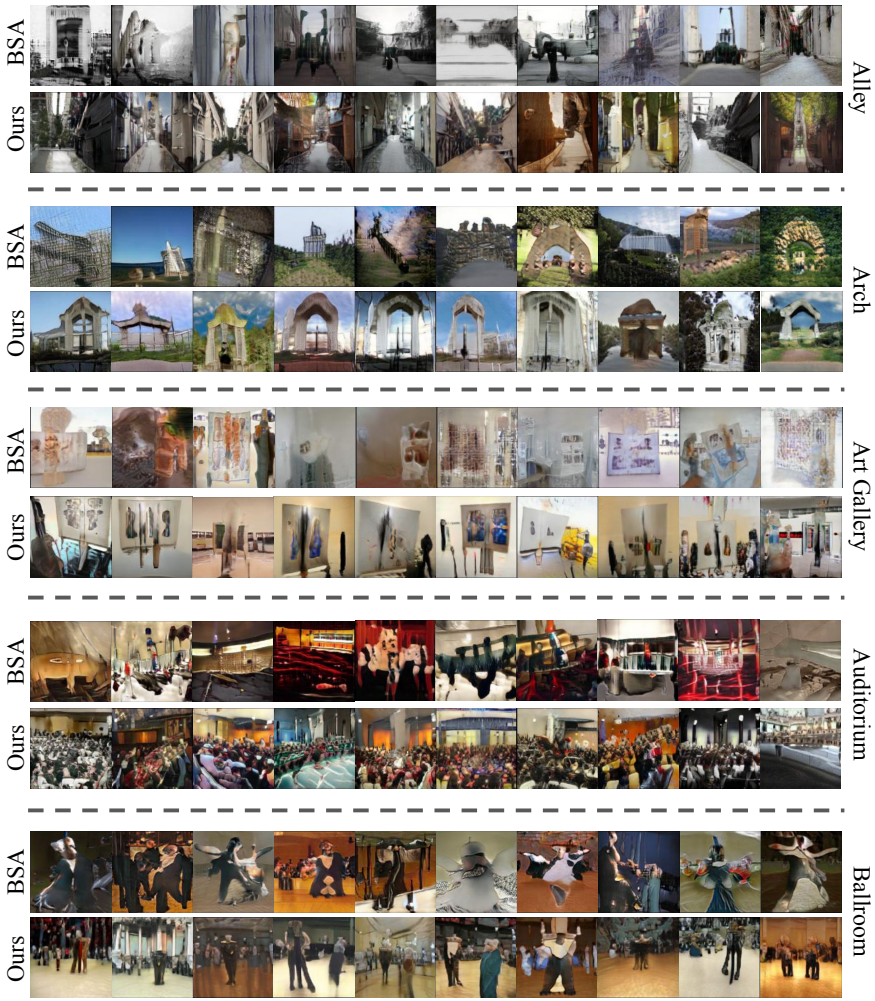


FIGURE A.5: Visual comparison between the images obtained from BSA (no knowledge transfer across classes) and our method on 5 classes of Places365.

FIGURE A.6: Visual comparison between the images obtained from BSA (no knowledge transfer across classes) and our method on some of the classes of Animal Face (for each class, the first row is from BSA, and the second row from our method).

FIGURE A.7: Visual comparison between the images obtained from BSA (left) and our method (right) for transferring from 80 classes of CIFAR100 to 20 classes each containing 300 samples.

FIGURE A.8: Visual comparison between the images obtained from BSA (left) and our method (right) for transferring from 80 classes of CIFAR100 to 10 classes each containing 300 samples.

B

APPENDIX: TRANSITIONAL CONDITIONING IN GANS

In the following, we provide more details, experiments, and visualizations on the discovered observation and the proposed method in chapter 3.

B.1 TRANSITION FUNCTION

In Fig. B.1, we provide a detailed visualization of the proposed transition function. As shown, T_s is the starting time and T_e is the end time of the transition. Note that the end of the training is different from the end of the transition. We have introduced a new term T_m in Fig. B.1 to represent the maximum training iterations. First, the model is trained unconditionally from $t = 0$ until $t = T_s$. At T_s, the transition to the condition model starts, λ_t going from 0 to 1 linearly. After the end of the transition (T_e), the transition function λ_t remains at its maximum value of 1. In other words, the weight of each loss in eq. (3.3) does not get adjusted anymore. The end of the transition T_e happens at about half the total training time T_m in our experiments.

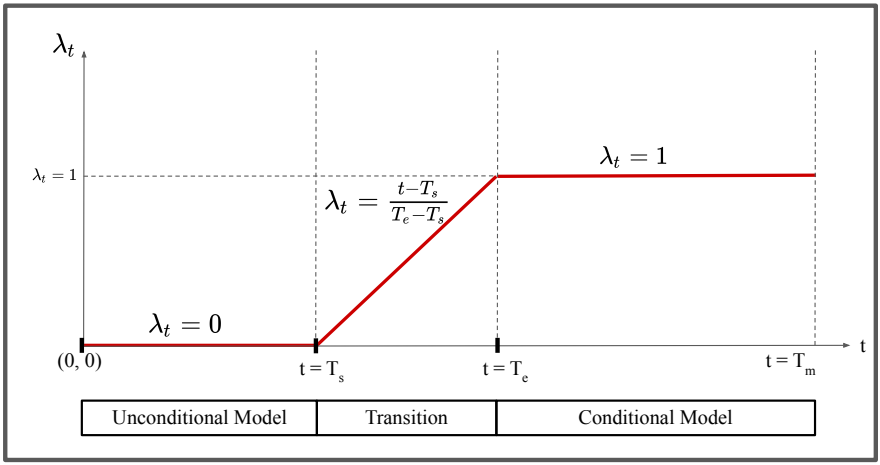


FIGURE B.1: Visualization of the transition function λ_t. T_s, T_e, and T_m denote the start of the transition, the end time of the transition, and the end of the training, respectively.

Method	C20, S500		C50, S50		C50, S300		C100, S300	
	FID	KID	FID	KID	FID	KID	FID	KID
UC-StyleGAN+ADA	**7**	**0.0006**	**20**	0.0022	**6**	**0.0008**	**6**	0.0021
C-StyleGAN+ADA	12	0.0033	23	0.0036	9	0.0025	13	0.0053
Ours	**7**	**0.0006**	**20**	**0.0014**	**6**	**0.0008**	**6**	**0.0012**

TABLE B.1: Quantitative results for examples of unconditional and conditional training of StyleGAN2+ADA, as well as our method, on different subsets of CIFAR100. In the name of the columns, C indicates the number of classes and S shows the number of images per class

B.2 MORE DETAILS ON THE IMPLEMENTATION

The proposed transition function makes a transition from 0 to 1 in the specified transition time. However, in the specific case of the AnimalFace dataset, we found that clipping the output of the transition function to the maximum value of 0.2 achieves the best results, which are reported in Sec. 3.4.2.

B.3 EXPERIMENTS ON CIFAR100

In addition to the experiments on the four already presented datasets, we provide the results of training unconditional and conditional StyleGAN2+ADA, as well as our method, on four different subsets of CIFAR100 ([89]) in Tab. B.1. In Fig. B.2, as an example, the FID curves of training the three methods on a subset of CIFAR100 with 100 classes and 300 images per class are visualized. We observe a similar behavior on CIFAR100, where our approach outperforms the conditional baseline, achieving FID and KID scores better than or on par with the unconditional baseline. .

B.4 CLASS-WISE FID AND KID

Following the common practice (StyleGAN, BigGAN, etc.), we calculate the FID and KID metrics reported in our experiments unconditionally, with a class sampling distribution that matches the class distribution of the training dataset. We did not provide the class-wise FID and KID due to the insufficient class-wise sample size. In Tab. B.2, we report the class-wise metrics for ImageNet Carnivores and Food101, by using all the images of corresponding classes, including the additional images not used for training.

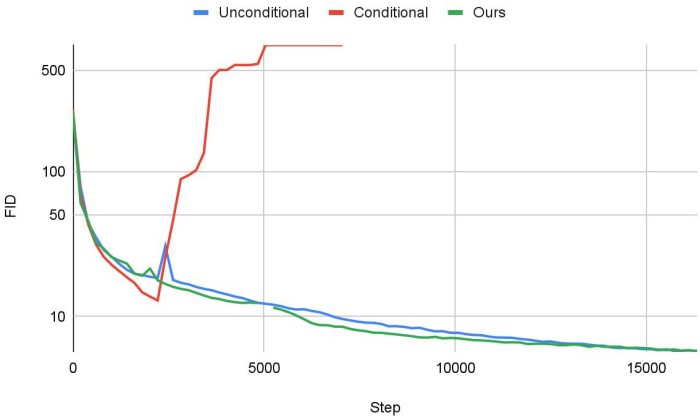


FIGURE B.2: FID curves for training unconditional and conditional StyleGAN2, as well as our method, on CIFAR100 with 100 classes and 300 images per class. The vertical axis is in the log scale.

Note that the values are generally larger for all methods, since a large fraction of FID/KID reference sets were not used for training. However, the relative values are still consistent with the metrics reported in Sec. 3.4.2. These measures, along with the generated images provided later in the appendix (Figs 9-12), show the class consistency of the proposed method.

B.5 PRECISION AND RECALL

In Tab. B.3, we provide the precision and recall proposed by [208], with the implementation provided by StyleGAN2+ADA. Based on the results, unconditional training always yields a higher recall, as it can generate between-mode images resulting in bigger diversity. Among the conditional methods, our method yields significantly better recall, while being comparable in terms of precision. Low recall values for the conditional baselines confirm the observed mode collapse. In Tab. B.4, we also provide the class-wise precision and recall for ImageNet Carnivores and Food101, calculated in the same manner as the class-wise FID and KID in appendix B.4.

Loss Formulation	Carnivores		Food101	
	FID	KID	FID	KID
C-StyleGAN+ADA	139	0.179	100	0.079
C-StyleGAN2+ADA+ProjD	151	0.199	97	0.0815
C-StyleGAN2+ADA+Lecam	90	0.096	56	0.027
Ours	**30**	**0.011**	**44**	**0.019**

TABLE B.2: The class-wise FID and KID for Imagenet Carnivores and Food101 using the full number of real samples per class in the evaluation.

Method	Carnivores		Food101		CUB-200-2011		AnimalFace	
	Pr	Rl	Pr	Rl	Pr	Rl	Pr	Rl
UC-StyleGAN2+ADA	0.77	0.300	0.71	0.187	0.70	0.232	0.76	0.430
C-StyleGAN2+ADA	0.80	0.0008	0.74	0.004	**0.79**	0.002	0.78	0.032
C-StyleGAN2+ADA+ProjD	0.81	0.0	0.74	0.002	0.76	0.067	**0.83**	0.0
C-StyleGAN2+ADA+Lecam	0.81	0.046	0.73	0.004	0.76	0.097	**0.83**	0.0005
Ours	**0.82**	**0.263**	**0.82**	**0.068**	0.77	**0.229**	**0.83**	**0.314**

TABLE B.3: The unconditional precision and recall for the compared methods.

B.6 MORE ABLATION: STYLEGAN2 WITHOUT ADA

To investigate whether the occurrence of the conditional collapse and efficacy of the proposed method in solving it is related to the adaptive differentiable augmentation (ADA), we perform further experiments on a subset of ImageNet carnivores (50 classes, 500 images per class), without using ADA in the training. As the FID curves in Fig. B.3 and FID and KID scores in Tab. B.5 show, the observed conditioning collapse happens even in the absence of ADA. Our method is still able to solve the problem by leveraging the stable behavior of unconditional training.

B.7 MORE ABLATION: TRANSITION IN THE LOSS FUNCTION

In Tab. B.6, we provide the ablation over two choices for transition in the proposed loss function:

- $L = (1 - \lambda_t) \cdot L_{uc} + \lambda_t \cdot L_c$.
- $L = L_{uc} + \lambda_t \cdot L_c$ (Ours).

Method	**Carnivores**		**Food101**	
	Pr	Rl	Pr	Rl
C-StyleGAN2+ADA	**0.75**	0.0	0.75	0
C-StyleGAN2+ADA+ProjD	0.68	0.0	**0.78**	0.00005
C-StyleGAN2+ADA+Lecam	0.67	0.0001	0.70	0.0148
Ours	0.73	**0.1205**	0.64	**0.1049**

TABLE B.4: The class-wise precision and recall for the compared methods on Imagenet Carnivores and Food101 using all of the available real samples per class in the evaluation.

Method	FID	KID
U-StyleGAN2	99	0.0795
C-StyleGAN2	13	0.0067
Ours	**8**	**0.0020**

TABLE B.5: The FID and KID scores for training unconditional and conditional StyleGAN2, as well as our method, without using ADA, on ImageNet Carnivores with 50 classes and 500 images per class.

The results show better performance for our proposed loss formulation. Based on the results, the unconditional and conditional training are not conflicting in the later stages of the training. Instead, having the unconditional loss helps the performance. However, as shown in Tab. 4.3, the two losses seem to be conflicting in the beginning of the training, resulting in a bad performance in the absence of the transition.

B.8 VISUAL RESULTS

In Fig. B.4, we provide visual results for the comparison of our method with the baselines. Moreover, more images randomly generated using our method on the four datasets are visualized in Figs. B.5 and B.8.

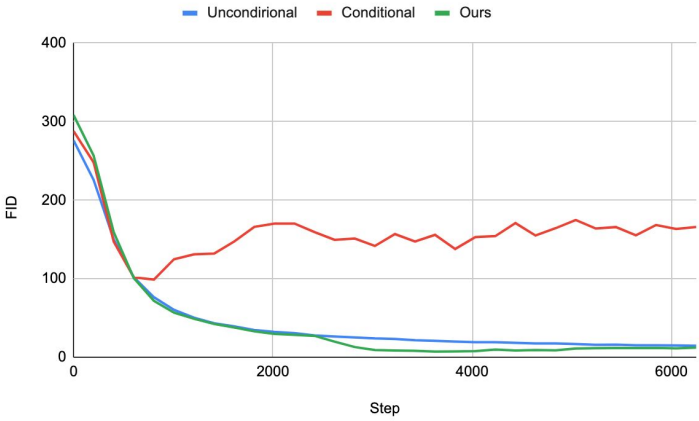


FIGURE B.3: FID curves for training unconditional and conditional StyleGAN2, as well as our method, on ImageNet Carnivores with 50 classes and 500 images per class.

Loss Formulation	**Carnivores**		**Food101**		**CUB-200-2011**		**AnimalFace**	
	FID	KID	FID	KID	FID	KID	FID	KID
$L = (1 - \lambda_t) \cdot L_{uc} + \lambda_t \cdot L_c$	18	0.0047	25	0.0088	25	0.0064	20	0.0035
$L = L_{uc} + \lambda_t \cdot L_c$ (Ours)	**14**	**0.0021**	**20**	**0.0045**	**22**	**0.0032**	**16**	**0.0018**

TABLE B.6: The FID and KID results for two different loss formulations.

FIGURE B.4: Visual results for the compared methods on four datasets.

FIGURE B.5: Randomly generated images using our method trained on ImageNet Carnivores dataset. Each row represents a different class. FID score is 14.

FIGURE B.6: Randomly generated images using our method trained on Food101 dataset. Each row represents a different class. The FID score is 20.

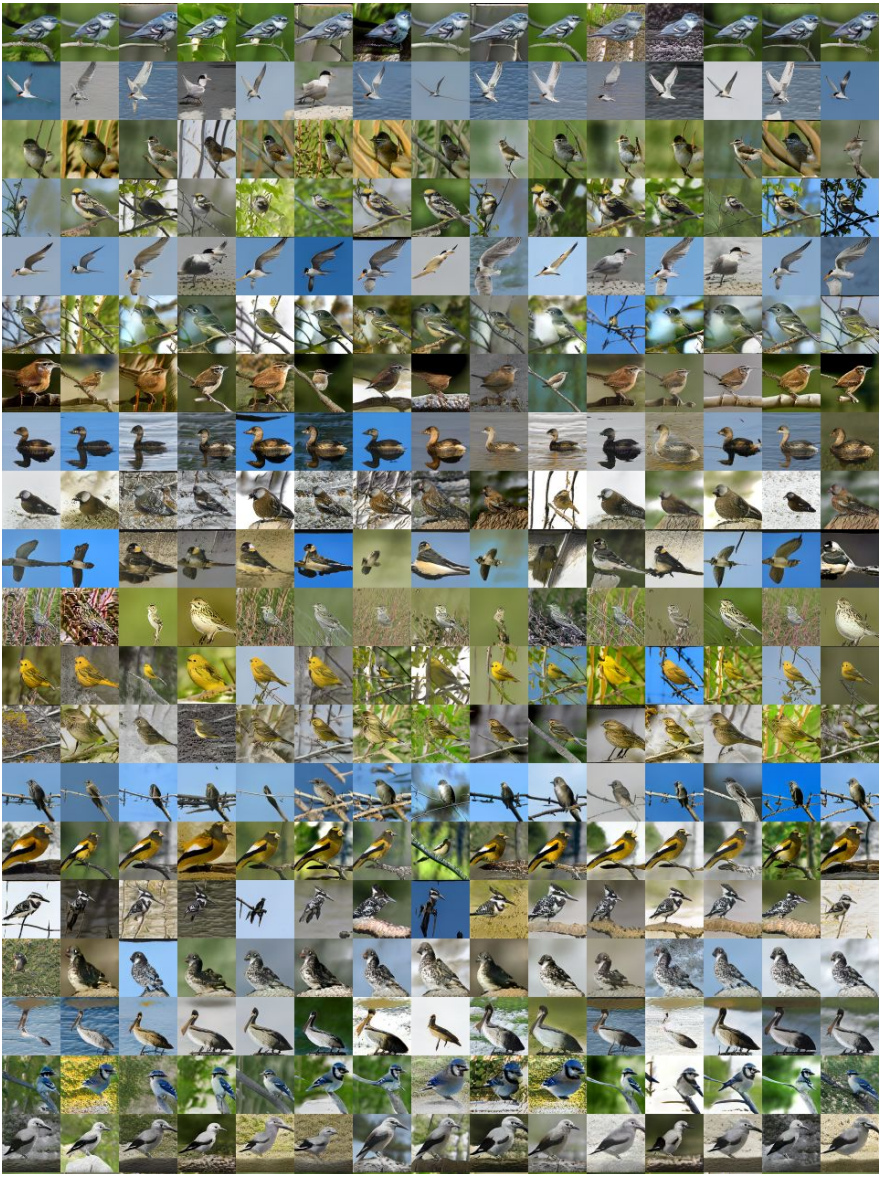


FIGURE B.7: Randomly generated images using our method trained on CUB-200-2011 dataset. Each row represents a different class. The FID score is 22.

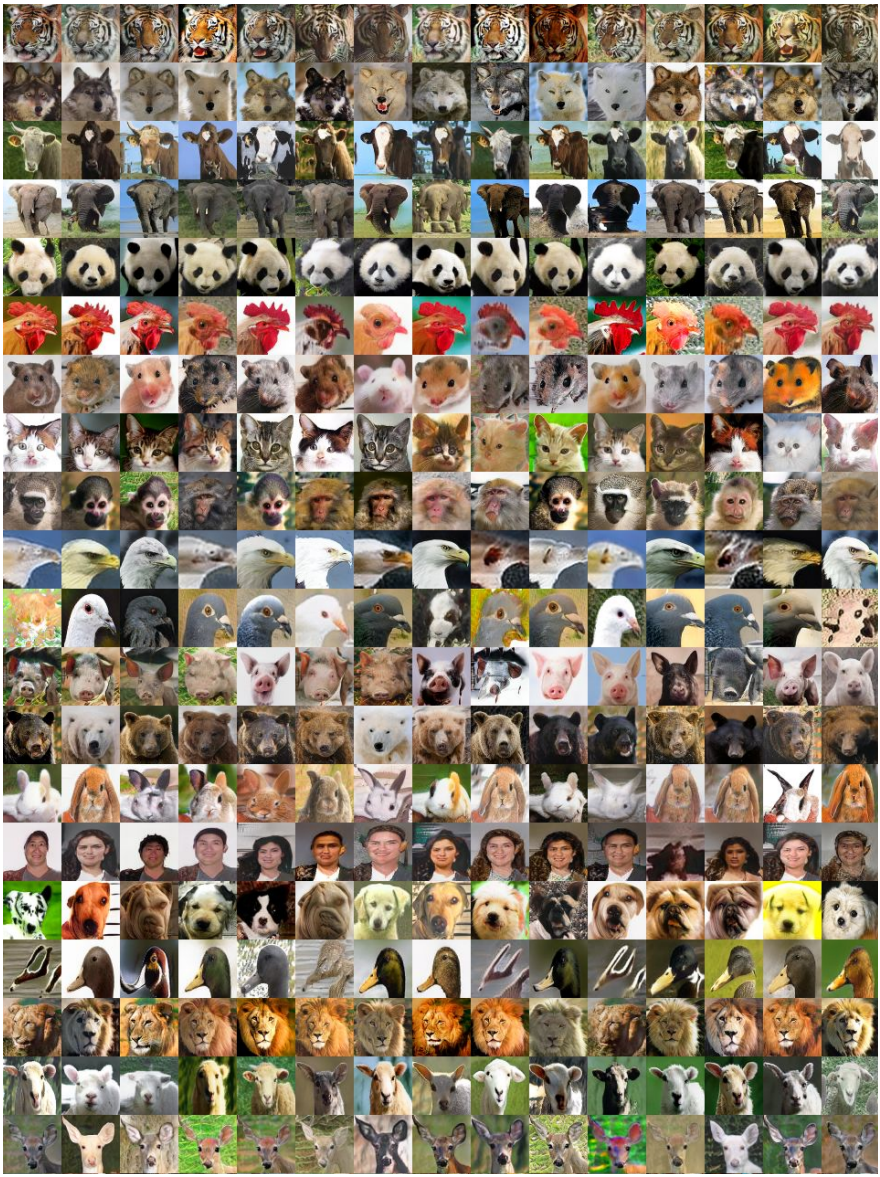


FIGURE B.8: Randomly generated images using our method trained on the AnimalFace dataset. Each row represents a different class. The FID score is 16.

C

APPENDIX: EFFICIENT 3D-AWARE GENERATION WITH CONVOLUTIONS

This appendix includes the additional material for chapter 4. In the following sections, we first provide an evaluation of the correspondence between the generated images using volumetric rendering and our convolutional generator in appendix C.1. Moreover, in appendix C.2, we extend the efficiency analysis of our method to the setup of the ShapeNet Cars dataset. In appendix C.3, we discuss some of the implementation details. Then, we offer a discussion on the limitations of the proposed method in appendix C.4. Finally, we provide more visual examples for our method in appendix C.5.

C.1 EVALUATION OF CORRESPONDENCE

To establish a baseline for future research, we measure the correspondence between generated images using our method and the corresponding images rendered from EG3D using Peak Signal-to-Noise Ratio (PSNR), the results of which are provided in Tab. C.1.

C.2 ANALYSIS OF EFFICIENCY ON SHAPENET CARS

In Fig. C.1, we provide a comparison of the computational efficiency of inference using our method and the baseline NeRF-GAN (EG3D) on ShapeNet Cars setup. EG3D performs volumetric rendering at the resolution of 64 and super-resolves the output to the resolution of 128. As seen in Fig. C.1, our method with StyleGAN2 as its backbone is more efficient than EG3D in terms of both memory consumption and speed. However, the StyleGAN3 backbone is only more efficient in terms of memory consumption, but it is slower than the baseline. This is because StyleGAN3 is more computationally demanding than the tri-plane generator in EG3D, which is based on StyleGAN2. In the setup of ShapeNet Cars, the computational complexity of the StyleGAN3 convolutional generator outweighs that of the lower-resolution volumetric rendering in EG3D, resulting in a slower inference (note that the memory consumption is still lower for the StyleGAN3 convolutional generator).

Dataset	ST2	ST3
FFHQ	28.73	28.65
AFHQ Cats	28.46	28.10
ShapeNet Cars	36.60	36.57

TABLE C.1: PSNR values (↑) measuring the correspondence between the images of EG3D and our convolutional generator on three datasets. The values are calculated for 1k images. ST2 and ST3 correspond to the two architectures StyleGAN2 and StyleGAN3 used as the backbone of our method

C.3 IMPLEMENTATION DETAILS

C.3.1 *SURF Baseline*

In Sec. 4.4, we provided a baseline, which we referred to as "SURF," inspired by the method proposed in the SURF-GAN paper [152] for pose/content disentanglement of a pre-trained 2D StyleGAN. As the code and the pre-trained models for such disentanglement are not made publicly available, we provided our best effort in implementing the SURF baseline inspired by their approach.

The proposed method in [152] consists of two main components: 1. a NeRF-GAN model called SURF-GAN for portrait images 2. a training recipe to add pose conditioning to a pre-trained StyleGAN2 using multi-view images generated from the 3D NeRF-GAN. Specifically, the part relevant as a baseline for our proposed method is the second component, which we re-implement for comparison with our method. To do so, for a fair comparison with our method, we use EG3D pre-trained on FFHQ for generating multi-view image triplets from 3 different viewpoints (source, canonical, and target views). Following [152], we use pSp [209], a pre-trained inversion encoder to invert the generated multi-view images in the style space of the 2D generator. Using the multi-view images and their corresponding style codes obtained using pSp, we train two mapping networks to:

1. map any arbitrary style code to the style code of the canonical viewpoint, and
2. map the canonical style code to the style code of the target viewpoint.

For the mapping networks, we use MLP networks with the same architecture as StyleGAN2's mapping network. For the training objective, we include reconstruction losses both on the images (MSE and Perceptual) and latent codes (MSE), as used in [152]. This architecture and training regime is inspired by the one in [152], but not exactly the same. We have adapted the method to the 3D generator network we are

using (EG3D). Moreover, in the original method, the mapping from the canonical latent code to the target one is formulated as the linear combination of a set of learnable pose vectors. In our implementation, we use a more general mapping by using an MLP network instead. Nevertheless, the implemented baseline provides a comparison with a proven alternative strategy to enable pose conditioning of a StyleGAN-like convolutional architecture.

C.3.2 *Inversion*

In Sec. 4.4.8, we provided examples of inversion in our generator using Pivotal Tuning Inversion (PTI) [159]. Following EG3D, inversion is performed given a target image and its camera parameters. For the implementation, we use the adaptation to EG3D provided at [210]. We use 500 steps for optimizing the latent code and 350 additional steps for fine-tuning the generator according to the PTI method.

C.3.3 *3D Consistency Metrics*

To calculate the 3D Landmark Consistency and Identity Preservation (ID) in Sec. 4.4.5.3, we vary the yaw from -40 to +40 and the pitch from -30 to +30, following the evaluation setup of [152]. As an additional visualization, Fig. C.2 shows the comparison of Identity Preservation for each angle individually.

C.4 LIMITATIONS

In our approach, the outputs of the volumetric rendering branch are used to supervise our convolutional renderer. Thus, the visual quality and 3D consistency of our approach is largely bound by the quality and consistency of the pre-trained NeRF-GAN. However, our formulation is largely agnostic to the volumetric generator used. Therefore, improvements in volumetric rendering in the context of GANs will also transfer to the generated quality of our method.

C.5 VISUAL RESULTS

In Fig. C.3, we provide more visual examples generated using our method from the FFHQ dataset. Fig. C.4 shows examples of generated images from the AFHQ Cats dataset using the proposed convolutional generator. Finally, in Fig. C.5, we provide additional samples for the ShapeNet Cars setup, generated using the volumetric rendering (EG3D) and our method.

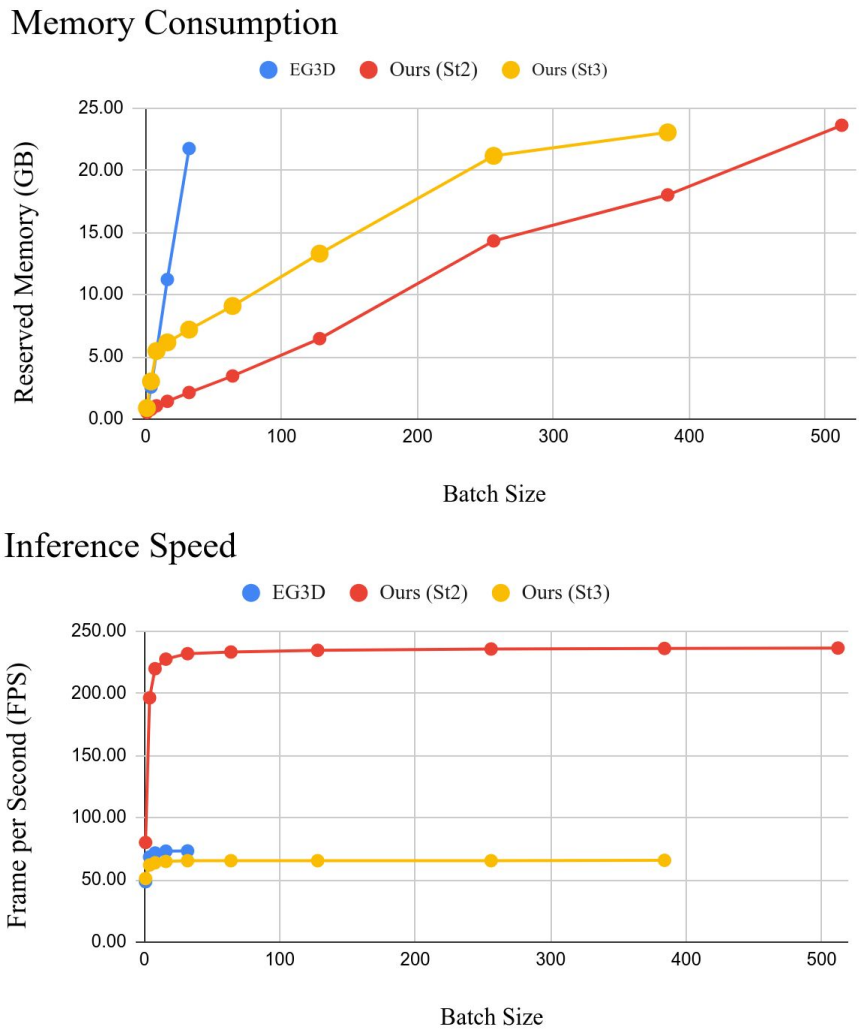


FIGURE C.1: Comparison of the inference memory consumption and speed (on a fixed GPU budget) for our method and EG3D on ShapeNet Cars.

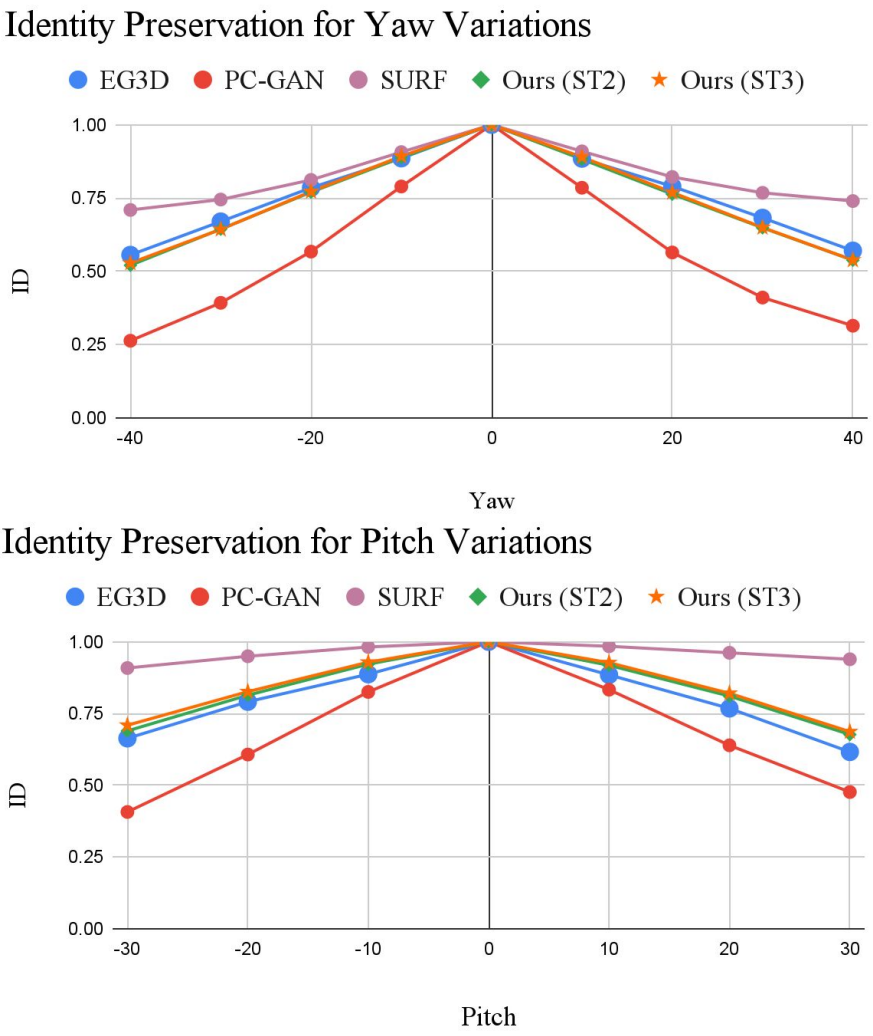


FIGURE C.2: Comparison of identity preservation across different camera poses. Our method performs much better than the pose-conditioned GAN baseline (PC-GAN), approaching the consistency of volumetric rendering (EG3D). The higher consistency achieved by SURF is due to its limited pose variations.

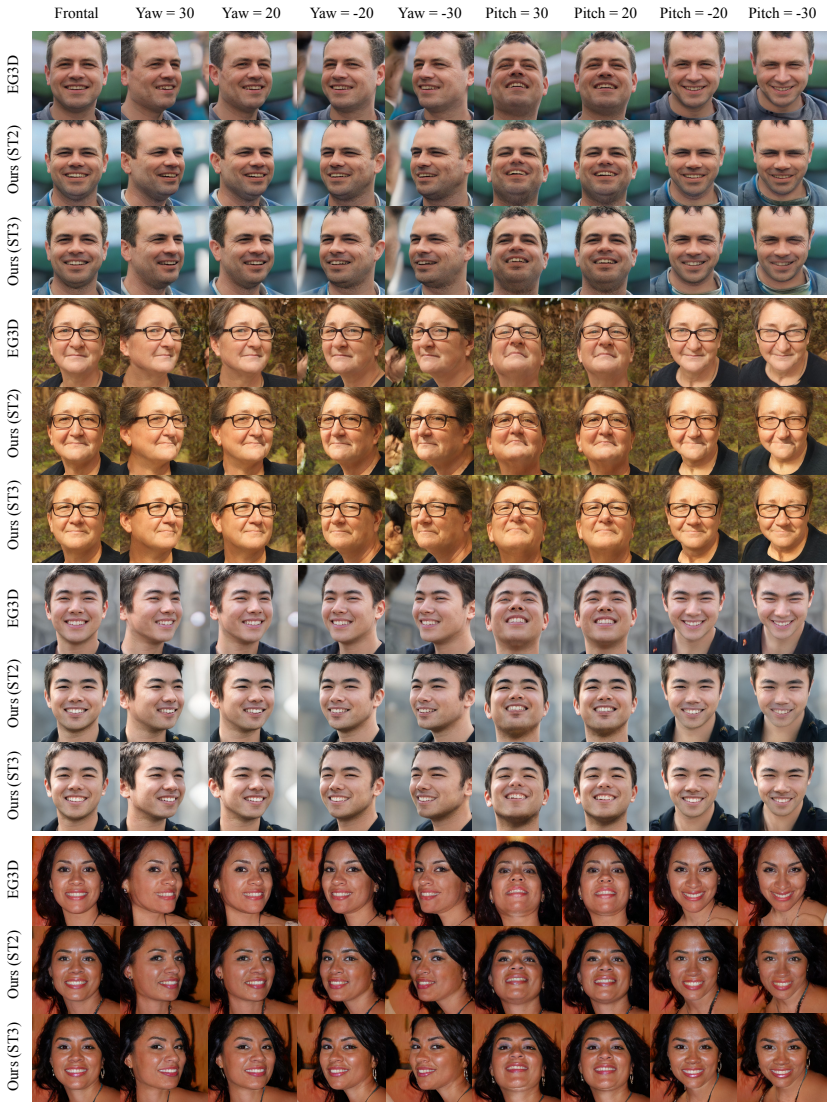


FIGURE C.3: Visual examples of pose control in our convolutional generator and their comparison to those of EG3D on FFHQ dataset.

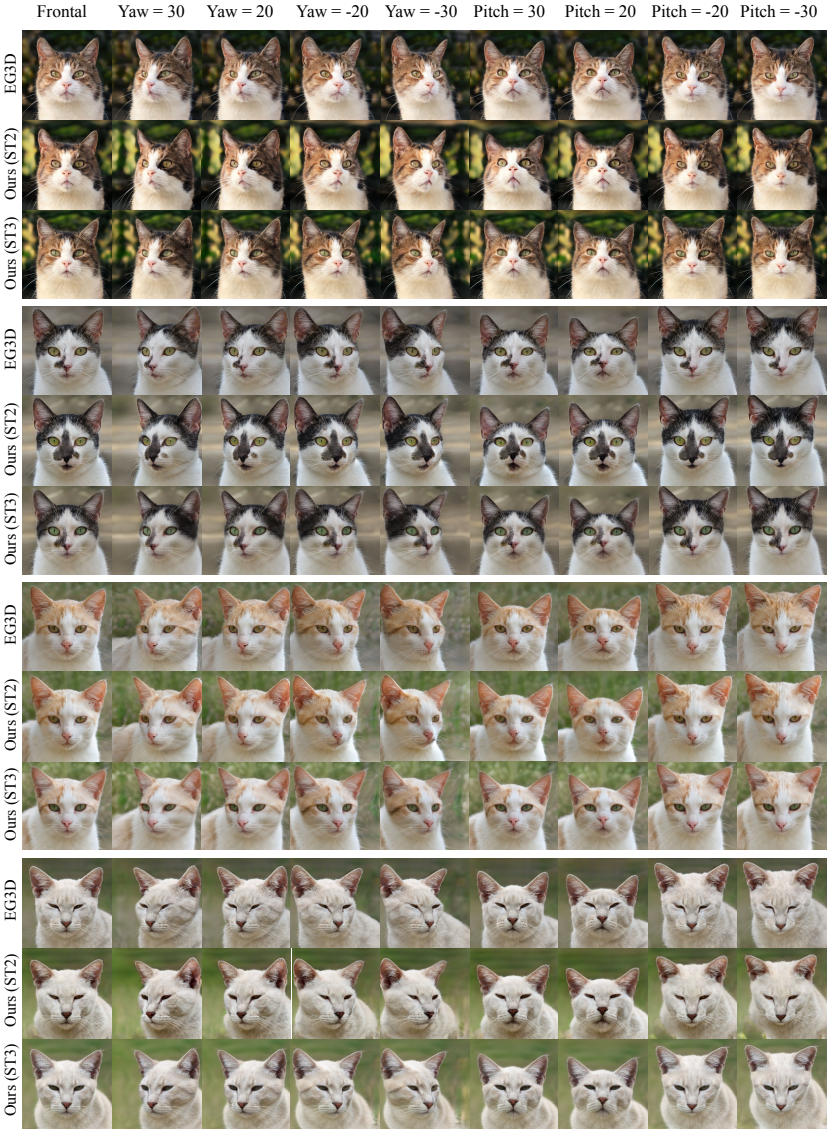


FIGURE C.4: Visual examples of pose control in our convolutional generator and their comparison to those of EG3D on the AFHQ Cats dataset.

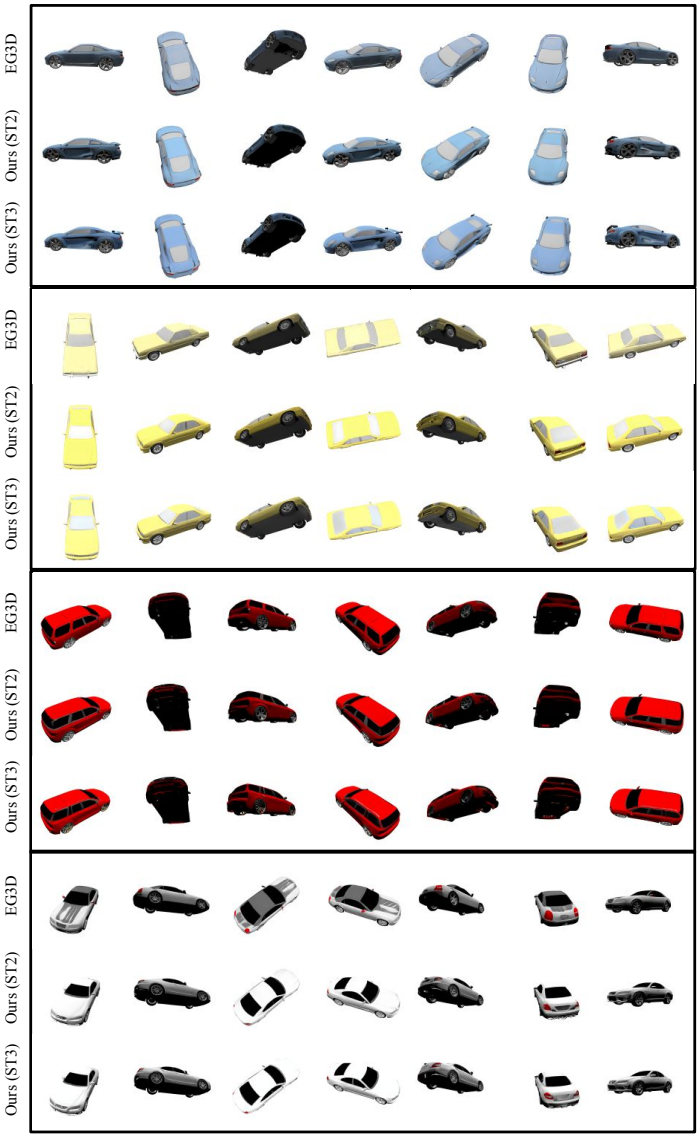


FIGURE C.5: Visual examples of pose control in our convolutional generator and their comparison to those of EG3D on the ShapeNet Cars dataset.

D

APPENDIX: GENERATIVE 3D OBJECT INSERTION

In this appendix, we provide additional visual results, a quantitative evaluation of our method, and an in-depth discussion of the implementation for the method proposed in chapter 5:

- appendix D.1 provides more visual examples of single and multiple object insertions using InseRF, additional visual comparisons with our baselines, more examples of the refinement step, as well as video visualizations.
- appendix D.2 offers more details on the components of our proposed method:
  - appendix D.2.1 provides more details on the bounding box-conditioned 2D editing of the reference view discussed in Sec. 5.3.2.
  - appendix D.2.2 provides a detailed description of the depth estimation and alignment proposed in Sec. 5.3.4.
  - appendix D.2.3 provides a detailed explanation on how we estimate the objects' rotation and translation in the 3D scenes in Sec. 5.3.4.
  - appendix D.2.4 offers additional details on the implementation of the object and scene fusion proposed in Sec. 5.3.5.
  - appendix D.2.5 provides a detailed explanation of our refinement step, proposed in Sec. 5.3.6, and its differences with the method proposed in Instruct-Nerf2Nerf [44].
- appendix D.3 offers a detailed discussion on the proposed baselines in Sec. 5.4.1: I-N2N [44], MV-Inpainting, Language-Driven Object Fusion [179].

D.1 ADDITIONAL VISUAL RESULTS

Visual examples: In Figs. 4.1 and 5.3, we provided examples of generative object insertion in 3D scenes using our proposed method. Here in Fig. D.1, we provide more visual examples showing the ability of our method to generate objects in 3D scenes.
Comparison to the baseline: In Fig. 5.4 , we provided visual comparisons between the proposed method and our baselines (introduced in Sec. 5.4.1). Fig. D.2 shows more comparisons with the baselines for a better assessment. As depicted, the two compared baselines struggle with creating the target objects in the scene.

Refinement: In Fig. 5.8 , we provided a visual ablation on the impact of the proposed refinement step in Sec. 5.3.6. Here in Fig. D.3, we extend the ablation to more examples. As can be seen, the proposed refinement step can improve the texture and details of the inserted objects, resulting in higher-quality and more realistic insertions. For the details of our refinement step, please refer to Sec. 5.3.6 and appendix D.2.5 of this supplementary.
Multiple object insertions: As shown in Fig. 5.9 , single object insertions using InseRF can be easily composited into multiple object insertions. More examples of such compositions are provided in Fig. D.4.
Video visualizations: To better visualize the inserted objects using our method, we additionally provide video visualizations in the supplementary files, showing several examples of our inserted objects, as well as examples of the refinement step and multiple object insertion. We additionally include a teaser video depicting how InseRF could be used to generate objects in 3D scenes. Please note that the text prompts shown in the teaser video are slightly modified for the sake of presentation. The exact prompts used for generating the examples are provided in Figs. 4.1 and 5.3 and Fig. D.1 in this supplementary document.

D.2 ADDITIONAL DETAILS ON THE METHOD

D.2.1 *Inpainting with RePaint*

As mentioned in Sec. 5.3.2, to generate a 2D view of the target object in the reference view, we condition our diffusion model on a bounding box using RePaint [183]. Repaint is a training-free inpainting method for pre-trained diffusion models that is capable of adding new content to an image in the regions specified by an arbitrary binary mask. Repaint primarily consists of 2 components: 1.) mask conditioning and 2.) re-sampling.

To enable mask conditioning, in every step t of the diffusion process, RePaint applies a mask-based blending to the output x_{t-1} as follows:

$$x_{t-1} = (1 - M) \odot x_{t-1}^{known} + M \odot x_{t-1}^{unknown} \tag{D.1}$$

where x_{t-1}^{known} is sampled using known pixels in the given image, $x_{t-1}^{unknown}$ is sampled from the model given the previous iteration x_t, and M is the binary mask. $\odot$ denotes element-wise multiplication. In our setup, we set M to be the area inside the condition bounding box and x_{t-1}^{known} to be noisy versions of the reference image x_0 obtained using the forward diffusion process.

When only applying the mask-based blending, the authors of RePaint observe that, although the inpainted region matches the texture of the neighboring region,

it is not well-harmonized in the image. Therefore, an additional re-sampling step is proposed, where the blended noisy images go through a few forward diffusion steps and are denoised again, to increase the harmonization of the inpainted regions. The proposed re-sampling step is characterized by two hyperparameters: 1) *jump length*: the number of applied forward diffusion steps, and 2) *steps*: the number of repetitions of adding noise and de-noising of the blended images. In our experiments, we set both parameters to the value 2.

D.2.2 *Depth Estimation*

As discussed in Sec. 5.3.4, we use the monocular depth estimated by MiDaS [189] to determine the location of the target object in the 3D frustum formed by the input bounding box in the reference image. As the estimated depth using MiDaS is non-metric, we perform a global affine depth alignment with the reference depth from the scene's NeRF reconstruction, which we explain in greater detail in the following.

Let D_R be the depth of the reference viewpoint rendered from the scene NeRF (not containing the object), and $\hat{D}_E$ be the estimated depth of the edited reference view (containing the 2D object) using MiDaS. We define the aligned depth map $\hat{D}_A$ of the edited reference view as:

$$\hat{D}_A = a \cdot \hat{D}_E + b, \tag{D.2}$$

where a and b are the scalar parameters of a global affine transformation. a and b are estimated by solving the following weighted least-square estimation:

$$\min_{a,b} \sum_i \sum_j (1 - M^{(i,j)}) \cdot W^{(i,j)} \cdot (D_R^{(i,j)} - \hat{D}_A^{(i,j)})^2, \tag{D.3}$$

where M is a binary mask corresponding to the reference bounding box. For a 2D matrix A, $A^{(i,j)}$ denotes the element at row i and column j. W is the matrix containing pixel-wise weights for the estimation, which correlate negatively with the distance of the pixel from the center of the bounding box located at row i_c and column j_c:

$$W_{ij} = 1 - \sqrt{(i - i_c)^2 + (j - j_c)^2}/z, \tag{D.4}$$

$$z = \max(\sqrt{(i - i_c)^2 + (j - j_c)^2}), \tag{D.5}$$

$$i \in \{0, ..., h-1\} \;\&\; j \in \{0, ..., w-1\}, \tag{D.6}$$

where z is a normalization term, and h and w are the height and width of the reference image, respectively. The weighted estimation of the alignment parameters

helps with a more accurate alignment in the region surrounding the inserted object. In practice, we perform our alignments on image crops containing the object and its surroundings instead of the full image.

After aligning the estimated depth map, in order to determine the location of the object in the 3D scene, we first roughly estimate the distance of the center of the object from the camera center to be equal to the depth value at the center of the bounding box d. Then, we perform the scale and distance optimization proposed in Sec. 5.3.4, with the constraint that the depth of the center of the object's rendered view from the reference viewpoint must be equal to d (please refer to the discussion on the scale and distance optimization in Sec. 5.3.4 for more details).

D.2.3 *Rotation and Translation*

Here we provide more details on the process of calculating the rotation and translation of the target object in the scene, discussed in Sec. 5.3.4. Specifically, we obtain the 3D location $\boldsymbol{p}_c$ of the center of the object in the 3D scene as the point along the normalized direction $\boldsymbol{v}$ pointing from the camera center to the center of the reference bounding box:

$$\boldsymbol{p}_c = \boldsymbol{o} + r^* \cdot \boldsymbol{v} \tag{D.7}$$

where r^* is the optimized distance obtained from the scale and radius optimization (explained in Sec. 5.3.4).

We use the right-handed coordinate system convention for our scene and object NeRFs and place the object in an upward position in the scene centered at $\boldsymbol{p}_c$. Moreover, we align the reference view of the object in its coordinate system (corresponding to zero azimuth and elevation) with the reference camera viewpoint in the scene's coordinate system. In other words, we define the axes of the object coordinate system in the scene's coordinate system as follows:

$$\boldsymbol{u}_{object} = [0, 0, 1]^T, \tag{D.8}$$

$$\boldsymbol{x}_{object} = -\boldsymbol{v}, \tag{D.9}$$

$$\boldsymbol{y}_{object} = normalize(\boldsymbol{u}_{object} \times \boldsymbol{x}_{object}), \tag{D.10}$$

$$\boldsymbol{z}_{object} = normalize(\boldsymbol{x}_{object} \times \boldsymbol{y}_{object}). \tag{D.11}$$

The rotation $\boldsymbol{R}$ and the translation $\boldsymbol{t}$ are then obtained as:

$$\boldsymbol{R} = [\boldsymbol{x}_{object}, \boldsymbol{y}_{object}, \boldsymbol{z}_{object}]^T \tag{D.12}$$

$$\boldsymbol{t} = -\boldsymbol{R}\boldsymbol{p}_c. \tag{D.13}$$

Using the obtained rotation, translation, and optimized object scale s^*, a point $\boldsymbol{p}$ in the scene's coordinate system can be mapped to a point $\boldsymbol{p'}$ in the object's one as follows:

$$\boldsymbol{p'} = \frac{1}{s^*}[\boldsymbol{R}, \boldsymbol{t}]\boldsymbol{p}. \tag{D.14}$$

D.2.4 *Scene and Object Fusion*

In Sec. 5.3.5, we provided a detailed discussion on how the scene and object NeRFs are fused in our method. In practice, object NeRFs fused in the scene may be queried with points in the 3D space that have not been seen during the object NeRF optimization, resulting in unwanted artifacts. To prevent such artifacts, we consider a 3D bounding box around the inserted objects, setting the density of the points sampled outside to zero. The dimensions of the 3D bounding box are determined based on the camera radius used in the single-view object reconstruction step and are fixed across edits and scenes.

D.2.5 *Refinement*

In Sec. 5.3.6, we proposed an optional refinement based on the iterative NeRF optimization proposed in Instruct-NeRF2NeRF with two modifications: 1) using the multi-view masks obtained from the inserted object to make the refinement localized and 2) sampling viewpoints on a sphere encapsulating the inserted object in the scene. In particular, we sample the viewpoints on a sphere with the radius r^* (the optimized object distance) from the object's center $\boldsymbol{p}_c$. Such a sampling strategy allows for better edits by the 2D diffusion model. Moreover, instead of randomly picking the next viewpoint to edit and include in the NeRF optimization, as done in Instruct-NeRF2NeRF, we order the viewpoints in such a way that more frontal views are selected first. For example, viewpoints $(azimuth, elevation)$ sampled from n equally-distanced azimuths with step size Δ_{theta} and m equally-distanced elevations with the step size Δ_{phi} are arranged as an ordered set V:

$$V = \{(i \cdot \Delta_\theta, j \cdot \Delta_\phi)) \mid i \in I \,\&\, j \in J\} \tag{D.15}$$

$$I = \{0, 1, -1..., n/2, -n/2\}, \tag{D.16}$$

$$J = \{0, 1, -1, ..., m/2, -m/2\}, \tag{D.17}$$

Such ordering improves the 3D consistency of the refinement step, as it decreases the conflict caused by randomly selected and independently edited views.

D.3 ADDITIONAL DETAILS ON BASELINES

Instruct-NeRF2NeRF (I-N2N): For our I-N2N baseline, we created a reimplementation in JAX on top of the Mip-NeRF360 code. Our implementation uses the official pre-trained checkpoints of Instruct-Pix2Pix [180] and is compatible with the LLFF datasets used in our experiments.

Multi-View Inpainting (MV-Inpainting): In Sec. 5.4.1, we proposed a baseline called Multi-View Inpainting (MV-Inpainting). MV-Inpainint is designed to insert objects into a 3D scene given accurate multi-view binary masks at the input. To ensure a fair comparison, MV-Inpainting uses the same 2D editing method as ours (Imagen [187] with RePaint [183]) to generate the target object in each viewpoint within the corresponding mask. In contrast to I-N2N, MV-Inpainting is equipped with localized editing to specifically investigate the importance of 3D consistency between different edited viewpoints.

To obtain the multi-view masks required for MV-Inpainting, we first generate and insert an object in the scene using our proposed object insertion. Then, we extract the multi-view masks of the target object by rendering the 3D object into the training viewpoints. The extracted masks are then used as input to MV-inpainting along with the corresponding text prompt. We would like to emphasize that our method only requires a single-view rough bounding box, in contrast to the multi-view accurate masks in MV-Inpainting.

Language-Driven Object Fusion: In Fig. 5.5 , we provided a preliminary comparison with the recent work, Language-Driven Object Fusion (LOF) [179] ([36]), introduced and explained in Secs. 5.2 and 5.4.1. LOF fuses an existing foreground object, represented by a set of multi-view images, into a background 3D scene conditioned on a user-provided 3D bounding box. To extend LOF to generate new objects in a 3D scene, first a 3D object based on a text prompt using DreamFusion [42]–a text-to-3D generative models–is generated. Then, their proposed method can be used to fuse the generated object with the scene based on the user-provided 3D bounding box, which determines the desired location, orientation and scale.

LOF is mainly designed and evaluated for fusing real objects with 3D scenes, which is different from and not comparable to our setup. The authors initially had further evaluated their proposed method on generated objects on two examples, which eventually was excluded in their final revision. We compare our method by applying the same prompt as that of one of those examples ("A sitting panda sculpture") to one of our own 3D scenes.

It is important to note that InseRF, different from LOF, does not rely on a 3D input bounding box. InseRF addresses a considerably more challenging task where the 3D placement is determined from only one single-view 2D bounding box from the user.

InseRF is more suitable for practical applications involving user interactions, as it is easier for the users to provide rough 2D bounding boxes rather than accurate 3D ones. Nevertheless, the comparison provided in Fig. 5.5 shows that InseRF achieves comparable results with LOF despite having no 3D guidance.

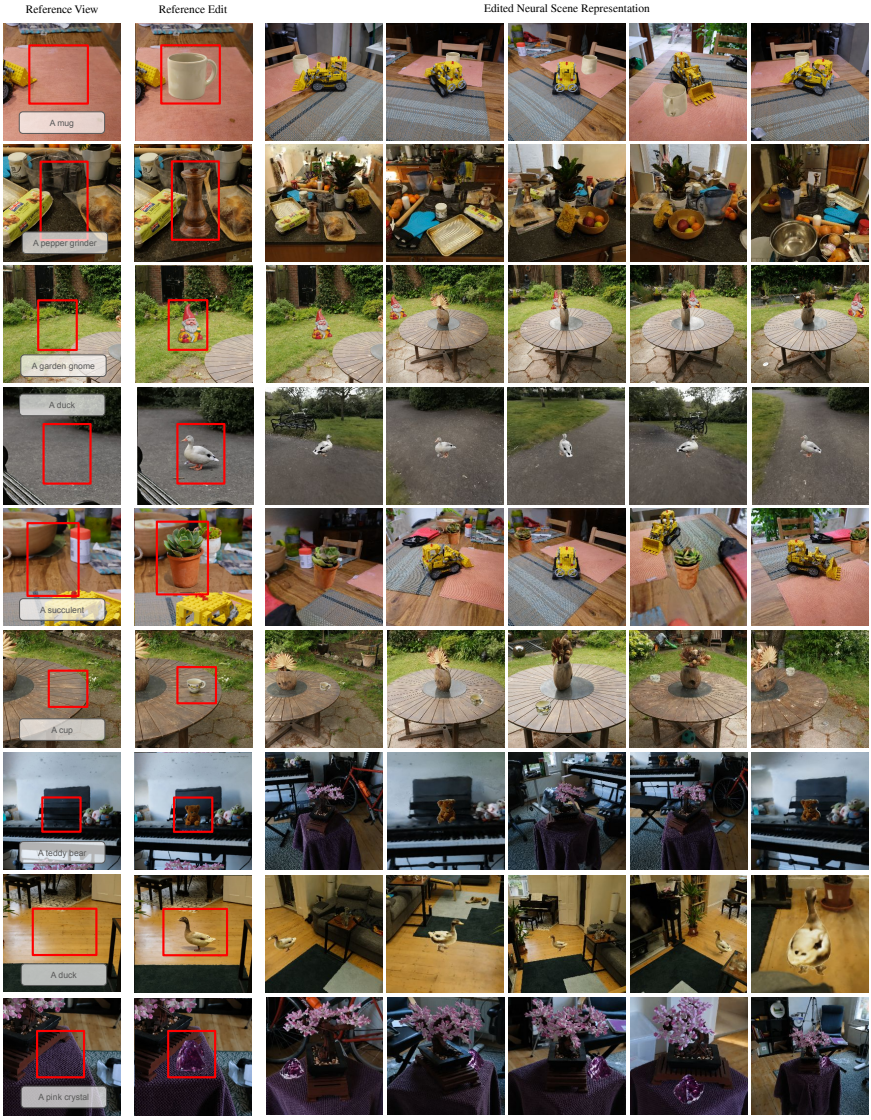
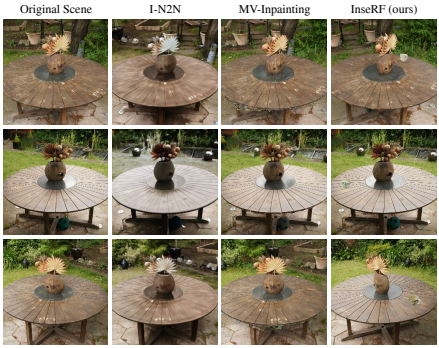
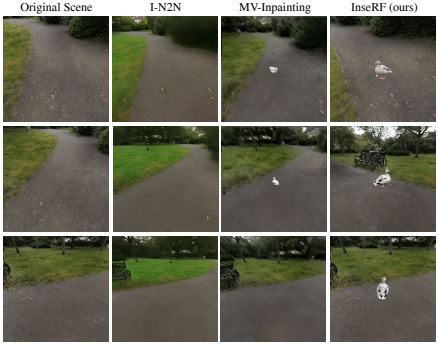


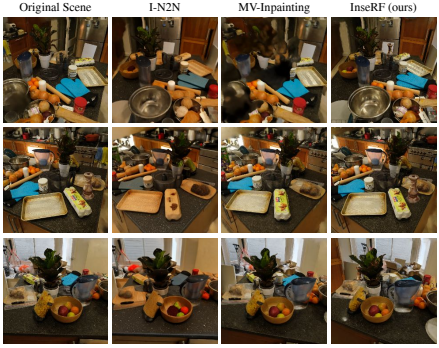
FIGURE D.1: Examples of using InseRF to insert an object into the neural representation of different indoor and outdoor scenes. More examples can be found in Fig. 5.3 .



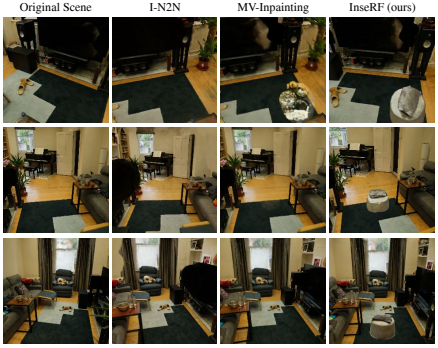
(a) A cup on the table



(b) A duck on the road



(c) A pepper grinder on the counter



(d) A pouffe on the carpet

FIGURE D.2: Qualitative comparison of object insertion with different methods. I-N2N and multiview inpainting both fail at inserting the geometry of the object at the desired location. Our method, in contrast, can insert new 3D-consistent objects at the desired location. More examples can be found in Fig. 5.4 .

(a) A mug (b) A wedding cake

FIGURE D.3: Additional visualizations of the effect of **refinement** (introduced in Sec. 5.3.6) on object insertion. Our refinement step can add additional texture details and lighting effects. More examples are provided in Fig. 5.8 .

(a) A panettone and a pepper grinder

(b) A cup, a wedding cake, and a garden gnome

FIGURE D.4: **Multiple object insertion.** Single inserted objects using our method can be arbitrarily composited into the scene, as also visualized in Fig. 5.9 .

BIBLIOGRAPHY

1. Zhang, Y., Ling, H., Gao, J., Yin, K., Lafleche, J.-F., Barriuso, A., Torralba, A. & Fidler, S. *DatasetGAN: Efficient Labeled Data Factory with Minimal Human Effort* in *CVPR* (2021).
2. Wu, W., Zhao, Y., Chen, H., Gu, Y., Zhao, R., He, Y., Zhou, H., Shou, M. Z. & Shen, C. DatasetDM: Synthesizing Data with Perception Annotations Using Diffusion Models. *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS 2023)* (2023).
3. He, R., Sun, S., Yu, X., Xue, C., Zhang, W., Torr, P., Bai, S. & QI, X. *IS SYNTHETIC DATA FROM GENERATIVE MODELS READY FOR IMAGE RECOGNITION?* in *The Eleventh International Conference on Learning Representations* (2023).
4. Tang, L., Jia, M., Wang, Q., Phoo, C. P. & Hariharan, B. *Emergent Correspondence from Image Diffusion* in *Thirty-seventh Conference on Neural Information Processing Systems* (2023).
5. Hedlin, E., Sharma, G., Mahajan, S., Isack, H., Kar, A., Tagliasacchi, A. & Yi, K. M. Unsupervised Semantic Correspondence Using Stable Diffusion (2023).
6. Li, D., Yang, J., Kreis, K., Torralba, A. & Fidler, S. *Semantic Segmentation With Generative Models: Semi-Supervised Learning and Strong Out-of-Domain Generalization* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), 8300.
7. Zhao, W., Rao, Y., Liu, Z., Liu, B., Zhou, J. & Lu, J. Unleashing Text-to-Image Diffusion Models for Visual Perception. *ICCV* (2023).
8. Ke, B., Obukhov, A., Huang, S., Metzger, N., Daudt, R. C. & Schindler, K. *Repurposing Diffusion-Based Image Generators for Monocular Depth Estimation* 2023.
9. Rabiner, L. & Juang, B. An introduction to hidden Markov models. *IEEE ASSP Magazine* **3**, 4 (1986).
10. McLachlan, G. J., Lee, S. X. & Rathnayake, S. I. Finite Mixture Models. *Annual Review of Statistics and Its Application* **6**, 355 (2019).
11. Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* (Morgan Kaufmann Publishers Inc., 1988).

12. Van den Oord, A., Kalchbrenner, N. & Kavukcuoglu, K. *Pixel Recurrent Neural Networks* in *Proceedings of The 33rd International Conference on Machine Learning* (eds Balcan, M. F. & Weinberger, K. Q.) **48** (PMLR, New York, New York, USA, 2016), 1747.

13. Van den Oord, A., Kalchbrenner, N., Espeholt, L., kavukcuoglu koray, k., Vinyals, O. & Graves, A. *Conditional Image Generation with PixelCNN Decoders* in *Advances in Neural Information Processing Systems* (eds Lee, D., Sugiyama, M., Luxburg, U., Guyon, I. & Garnett, R.) **29** (Curran Associates, Inc., 2016).

14. Kingma, D. P. & Welling, M. *Auto-Encoding Variational Bayes* 2022.

15. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. *Generative adversarial nets* in *Advances in Neural Information Processing Systems* (2014), 2672.

16. Rezende, D. & Mohamed, S. *Variational Inference with Normalizing Flows* in *Proceedings of the 32nd International Conference on Machine Learning* (eds Bach, F. & Blei, D.) **37** (PMLR, Lille, France, 2015), 1530.

17. Ho, J., Jain, A. & Abbeel, P. Denoising Diffusion Probabilistic Models. *arXiv preprint arxiv:2006.11239* (2020).

18. Brock, A., Donahue, J. & Simonyan, K. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096* (2018).

19. Karras, T., Laine, S. & Aila, T. *A style-based generator architecture for generative adversarial networks* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2019), 4401.

20. Isola, P., Zhu, J.-Y., Zhou, T. & Efros, A. A. Image-to-Image Translation with Conditional Adversarial Networks. *CVPR* (2017).

21. Park, T., Liu, M.-Y., Wang, T.-C. & Zhu, J.-Y. *Semantic Image Synthesis with Spatially-Adaptive Normalization* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

22. Zhu, J.-Y., Park, T., Isola, P. & Efros, A. A. *Unpaired image-to-image translation using cycle-consistent adversarial networks* in *Proceedings of the IEEE international conference on computer vision* (2017), 2223.

23. Ling, H., Kreis, K., Li, D., Kim, S. W., Torralba, A. & Fidler, S. *EditGAN: High-Precision Semantic Image Editing* in *Advances in Neural Information Processing Systems (NeurIPS)* (2021).

24. Sauer, A., Karras, T., Laine, S., Geiger, A. & Aila, T. *StyleGAN-T: Unlocking the Power of GANs for Fast Large-Scale Text-to-Image Synthesis* in. **abs/2301.09515** (2023).

25. Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J. & Aila, T. *Training Generative Adversarial Networks with Limited Data* in *Advances in Neural Information Processing Systems* (2020).

26. Rombach, R., Blattmann, A., Lorenz, D., Esser, P. & Ommer, B. *High-Resolution Image Synthesis with Latent Diffusion Models* 2021.

27. Chen, S., Sun, P., Song, Y. & Luo, P. DiffusionDet: Diffusion Model for Object Detection. *arXiv preprint arXiv:2211.09788* (2022).

28. Song, J., Meng, C. & Ermon, S. Denoising Diffusion Implicit Models. *arXiv:2010.02502* (2020).

29. Song, Y., Dhariwal, P., Chen, M. & Sutskever, I. Consistency Models. *arXiv preprint arXiv:2303.01469* (2023).

30. Xiao, Z., Kreis, K. & Vahdat, A. *Tackling the Generative Learning Trilemma with Denoising Diffusion GANs* in *International Conference on Learning Representations (ICLR)* (2022).

31. Sauer, A., Lorenz, D., Blattmann, A. & Rombach, R. *Adversarial Diffusion Distillation* 2023.

32. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J. & Aila, T. *Analyzing and Improving the Image Quality of StyleGAN* in *Proc. CVPR* (2020).

33. Hatamizadeh, A., Song, J., Liu, G., Kautz, J. & Vahdat, A. *DiffiT: Diffusion Vision Transformers for Image Generation* 2023.

34. Chan, E. R., Lin, C. Z., Chan, M. A., Nagano, K., Pan, B., De Mello, S., Gallo, O., Guibas, L. J., Tremblay, J., Khamis, S., *et al. Efficient geometry-aware 3D generative adversarial networks* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), 16123.

35. Pan, S. J. & Yang, Q. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering* **22**, 1345 (2010).

36. Krizhevsky, A., Sutskever, I. & Hinton, G. E. *ImageNet Classification with Deep Convolutional Neural Networks* in *Advances in Neural Information Processing Systems* (eds Pereira, F., Burges, C., Bottou, L. & Weinberger, K.) **25** (Curran Associates, Inc., 2012).

37. Snell, J., Swersky, K. & Zemel, R. *Prototypical networks for few-shot learning* in *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Curran Associates Inc., Long Beach, California, USA, 2017), 4080.

38. Finn, C., Abbeel, P. & Levine, S. *Model-agnostic meta-learning for fast adaptation of deep networks* in *Proceedings of the 34th International Conference on Machine Learning - Volume 70* (JMLR.org, Sydney, NSW, Australia, 2017), 1126.

39. Hinton, G. E., Vinyals, O. & Dean, J. Distilling the Knowledge in a Neural Network. *ArXiv* **abs/1503.02531** (2015).

40. Frankle, J. & Carbin, M. *The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks* in *International Conference on Learning Representations* (2019).

41. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H. & Kalenichenko, D. *Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

42. Poole, B., Jain, A., Barron, J. T. & Mildenhall, B. DreamFusion: Text-to-3D using 2D Diffusion. *arXiv* (2022).

43. Liu, R., Wu, R., Hoorick, B. V., Tokmakov, P., Zakharov, S. & Vondrick, C. *Zero-1-to-3: Zero-shot One Image to 3D Object* 2023.

44. Haque, A., Tancik, M., Efros, A., Holynski, A. & Kanazawa, A. *Instruct-NeRF2NeRF: Editing 3D Scenes with Instructions* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023).

45. Dong, J. & Wang, Y.-X. *ViCA-NeRF: View-Consistency-Aware 3D Editing of Neural Radiance Fields* in *Thirty-seventh Conference on Neural Information Processing Systems* (2023).

46. Zhuang, J., Wang, C., Liu, L., Lin, L. & Li, G. DreamEditor: Text-Driven 3D Scene Editing with Neural Fields. *arXiv preprint arXiv:2306.13455* (2023).

47. Mirzaei, A., Aumentado-Armstrong, T., Brubaker, M. A., Kelly, J., Levinshtein, A., Derpanis, K. G. & Gilitschenski, I. *Reference-guided Controllable Inpainting of Neural Radiance Fields* in *ICCV* (2023).

48. Wang, D., Zhang, T., Abboud, A. & Süsstrunk, S. InpaintNeRF360: Text-Guided 3D Inpainting on Unbounded Neural Radiance Fields. *arXiv preprint arXiv:2305.15094* (2023).

49. Arjovsky, M., Chintala, Soumith & Bottou, L. Wasserstein Generative Adversarial Networks. *arXiv preprint arXiv:1701.07875* (2017).

50. Liu, M.-Y., Huang, X., Yu, J., Wang, T.-C. & Mallya, A. Generative Adversarial Networks for Image and Video Synthesis: Algorithms and Applications. *arXiv preprint arXiv:2008.02793* (2020).

51. Karras, T., Aila, T., Laine, S. & Lehtinen, J. *Progressive Growing of GANs for Improved Quality, Stability, and Variation* in *International Conference on Learning Representations* (2018).

52. Wu, J., Huang, Z., Acharya, D., Li, W., Thoma, J., Paudel, D. P. & Gool, L. V. *Sliced wasserstein generative models* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), 3713.

53. Park, T., Liu, M.-Y., Wang, T.-C. & Zhu, J.-Y. *Semantic image synthesis with spatially-adaptive normalization* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), 2337.

54. Clark, A., Donahue, J. & Simonyan, K. Adversarial video generation on complex datasets. *arXiv preprint arXiv:1907.06571* (2019).

55. Pan, S. J. & Yang, Q. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* **22**, 1345 (2009).

56. Zamir, A. R., Sax, A., Shen, W., Guibas, L. J., Malik, J. & Savarese, S. *Taskonomy: Disentangling task transfer learning* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), 3712.

57. Ganin, Y. & Lempitsky, V. *Unsupervised domain adaptation by backpropagation* in *International conference on machine learning* (2015), 1180.

58. Liu, M.-Y. & Tuzel, O. *Coupled generative adversarial networks* in *Advances in Neural Information Processing Systems* (2016), 469.

59. Tzeng, E., Hoffman, J., Saenko, K. & Darrell, T. *Adversarial discriminative domain adaptation* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), 7167.

60. Wang, Y., Wu, C., Herranz, L., van de Weijer, J., Gonzalez-Garcia, A. & Raducanu, B. *Transferring GANs: generating images from limited data* in *Proceedings of the European Conference on Computer Vision* (2018), 218.

61. Mo, S., Cho, M. & Shin, J. Freeze Discriminator: A Simple Baseline for Fine-tuning GANs. *arXiv preprint arXiv:2002.10964* (2020).

62. Wang, Y., Gonzalez-Garcia, A., Berga, D., Herranz, L., Khan, F. S. & Weijer, J. v. d. *MineGAN: effective knowledge transfer from GANs to target domains with few images* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), 9332.

63. Zhao, M., Cong, Y. & Carin, L. On Leveraging Pretrained GANs for Limited-Data Generation. *arXiv preprint arXiv:2002.11810* (2020).

64. Noguchi, A. & Harada, T. *Image generation from small datasets via batch statistics adaptation* in *Proceedings of the IEEE International Conference on Computer Vision* (2019), 2750.

65. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. & Fei-Fei, L. *Imagenet: A large-scale hierarchical image database* in *IEEE conference on computer vision and pattern recognition* (2009), 248.

66. Li, Z. & Hoiem, D. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence* **40**, 2935 (2017).

67. Parisi, G. I., Kemker, R., Part, J. L., Kanan, C. & Wermter, S. Continual lifelong learning with neural networks: A review. *Neural Networks* **113**, 54 (2019).

68. Cong, Y., Zhao, M., Li, J., Wang, S. & Carin, L. *GAN Memory with No Forgetting* in *Advances in Neural Information Processing Systems* (2020), 469.

69. Bengio, Y., Louradour, J., Collobert, R. & Weston, J. *Curriculum learning* in *Proceedings of the 26th annual international conference on machine learning* (2009), 41.

70. Hinton, G., Vinyals, O. & Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

71. Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., *et al. Matching networks for one shot learning* in *Advances in Neural Information Processing Systems* (2016), 3630.

72. Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H. & Hospedales, T. M. *Learning to compare: Relation network for few-shot learning* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), 1199.

73. Guo, Y., Ding, G., Gao, Y. & Han, J. *Active Learning with Cross-Class Similarity Transfer.* in *Association for the Advancement of Artificial Intelligence* (2017), 1338.

74. Data, G. W. P., Ngu, K., Murray, D. W. & Prisacariu, V. A. *Interpolating convolutional neural networks using batch normalization* in *Proceedings of the European Conference on Computer Vision* (2018).

75. Panareda Busto, P. & Gall, J. *Open set domain adaptation* in *Proceedings of the IEEE International Conference on Computer Vision* (2017), 754.

76. Zhou, F., Jiang, Z., Shui, C., Wang, B. & Chaib-draa, B. Domain Generalization with Optimal Transport and Metric Learning. *arXiv preprint arXiv:2007.10573* (2020).

77. Liu, Z., Miao, Z., Pan, X., Zhan, X., Lin, D., Yu, S. X. & Gong, B. *Open Compound Domain Adaptation* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020).

78. Mirza, M. & Osindero, S. Conditional Generative Adversarial Nets. *arXiv preprint arXiv:1411.1784* (2014).

79. Miyato, T. & Koyama, M. cGANs with Projection Discriminator. *arXiv preprint arXiv:1802.05637* (2018).

80. Kavalerov, I., Czaja, W. & Chellappa, R. cGANs with Multi-Hinge Loss. *arXiv preprint arXiv:1912.04216* (2019).

81. Miyato, T., Kataoka, T., Koyama, M. & Yoshida, Y. cGANs with Projection Discriminator. *arXiv preprint arXiv:1802.05957* (2018).

82. De Vries, H., Strub, F., Mary, J., Larochelle, H., Pietquin, O. & Courville, A. *Modulating early visual processing by language* in *Advances in Neural Information Processing Systems* (2017), 6594.

83. Zhang, H., Goodfellow, I., Metaxas, D. & Odena, A. *Self-attention generative adversarial networks* in *International conference on machine learning* (2019), 7354.

84. Nguyen, A., Dosovitskiy, A., Yosinski, J., Brox, T. & Clune, J. *Synthesizing the preferred inputs for neurons in neural networks via deep generator networks* in *Advances in Neural Information Processing Systems* (2016), 3387.

85. Nguyen, A., Clune, J., Bengio, Y., Dosovitskiy, A. & Yosinski, J. *Plug & play generative networks: Conditional iterative generation of images in latent space* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), 4467.

86. Si, Z. & Zhu, S.-C. Learning Hybrid Image Templates (HIT) by Information Projection. *IEEE Transactions on pattern analysis and machine intelligence* **34**, 1354 (2011).

87. Zhao, S., Liu, Z., Lin, J., Zhu, J.-Y. & Han, S. *Differentiable augmentation for data-efficient gan training* in *Advances in Neural Information Processing Systems* **33** (2020).

88. lim, J. H. & Ye, J. C. Geometric GAN. *arXiv preprint arXiv:1705.02894* (2017).

89. Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images. *Tech Report* (2009).

90. Zhou, B., Lapedriza, A., Xiao, J., Torralba, A. & Oliva, A. *Learning deep features for scene recognition using places database* in *Advances in Neural Information Processing Systems* (2014), 487.

91. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B. & Hochreiter, S. *Gans trained by a two time-scale update rule converge to a local nash equilibrium* in *Advances in Neural Information Processing Systems* (2017), 6626.

92. Mirza, M. & Osindero, S. Conditional Generative Adversarial Nets. *arXiv preprint arXiv:1411.1784* (2014).

93. Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X. & Huang, T. S. *Generative image inpainting with contextual attention* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), 5505.

94. Xu, T., Zhang, P., Huang, Q., Zhang, H., Gan, Z., Huang, X. & He, X. *Attngan: Fine-grained text to image generation with attentional generative adversarial networks* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), 1316.

95. Tseng, H.-Y., Jiang, L., Liu, C., Yang, M.-H. & Yang, W. *Regularing Generative Adversarial Networks under Limited Data* in *CVPR* (2021).

96. Gupta, A., Dollar, P. & Girshick, R. *LVIS: A dataset for large vocabulary instance segmentation* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), 5356.

97. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Alec, R. & Xi, C. *Improved Techniques for Training GANs* in *Advances in Neural Information Processing Systems* (2016).

98. Zhou, Z., Cai, H., Rong, S., Song, Y., Ren, K., Zhang, W., Wang, J. & Yu, Y. *Activation Maximization Generative Adversarial Nets* in *International Conference on Learning Representations* (2018).

99. Kavalerov, I., Czaja, W. & Chellappa, R. *A study of quality and diversity in K+1 GANs* in *Proceedings on "I Can't Believe It's Not Better!" at NeurIPS Workshops* (2020).

100. Liu, M.-Y., Huang, X., Mallya, A., Karras, T., Aila, T., Lehtinen, J. & Kautz, J. *Few-Shot Unsupervised Image-to-Image Translation* in *IEEE International Conference on Computer Vision (ICCV)* (2019).

101. Miyato, T. & Koyama, M. *cGANs with Projection Discriminator* in *International Conference on Learning Representations* (2018).

102. Wah, C., Branson, S., Welinder, P., Perona, P. & Belongie, S. *The Caltech-UCSD Birds-200-2011 Dataset* tech. rep. CNS-TR-2011-001 (California Institute of Technology, 2011).

103. Bossard, L., Guillaumin, M. & Van Gool, L. *Food-101 – Mining Discriminative Components with Random Forests* in *European Conference on Computer Vision* (2014).

104. Bińkowski, M., Sutherland, D. J., Arbel, M. & Gretton, A. *Demystifying MMD GANs* in *International Conference on Learning Representations* (2018).

105. Kang, M. & Park, J. *ContraGAN: Contrastive Learning for Conditional Image Generation* in *Conference on Neural Information Processing Systems (NeurIPS)* (2020).

106. Odena, A., Olah, C. & Shlens, J. *Conditional Image Synthesis with Auxiliary Classifier GANs* in *International conference on machine learning* (2017), 2642.

107. Shahbazi, M., Huang, Z., Paudel, D. P., Chhatkuli, A. & Van Gool, L. *Efficient Conditional GAN Transfer with Knowledge Propagation across Classes* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021).

108. Liu, B., Zhu, Y., Song, K. & Elgammal, A. *Towards Faster and Stabilized {GAN} Training for High-fidelity Few-shot Image Synthesis* in *Submitted to International Conference on Learning Representations* (2021).

109. Ramasinghe, S., Farazi, M., Khan, S., Barnes, N. & Gould, S. *Rethinking conditional GAN training: An approach using geometrically structured latent manifolds* in *NeurIPS* (2021).

110. Lugmayr, A., Danelljan, M., Yu, F., Van Gool, L. & Timofte, R. *Normalizing Flow as a Flexible Fidelity Objective for Photo-Realistic Super-Resolution* in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)* (2022), 1756.

111. Lee, S., Ha, J. & Kim, G. *Harmonizing Maximum Likelihood with GANs for Multimodal Conditional Generation* in *International Conference on Learning Representations* (2019).

112. Karras, T., Aila, T., Laine, S. & Lehtinen, J. *Progressive Growing of GANs for Improved Quality, Stability, and Variation* in *Proceedings of the International Conference on Learning Representations (ICLR)* (2018).

113. Choi, Y., Uh, Y., Yoo, J. & Ha, J.-W. *StarGAN v2: Diverse Image Synthesis for Multiple Domains* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2020).

114. Sauer, A., Schwarz, K. & Geiger, A. *StyleGAN-XL: Scaling StyleGAN to Large Diverse Datasets* in. **abs/2201.00273** (2022).

115. Sauer, A., Chitta, K., Muller, J. & Geiger, A. *Projected GANs Converge Faster* in *NeurIPS* (2021).

116. Nguyen-Phuoc, T., Li, C., Theis, L., Richardt, C. & Yang, Y.-L. *HoloGAN: Unsupervised Learning of 3D Representations From Natural Images* in *The IEEE International Conference on Computer Vision (ICCV)* (2019).

117. Tewari, A., Elgharib, M., Bharaj, G., Bernard, F., Seidel, H.-P., Perez, P., Zollhofer, M. & Theobalt, C. *StyleRig: Rigging StyleGAN for 3D Control Over Portrait Images* in (2020), 6141.

118. Kowalski, M., Garbin, S. J., Estellers, V., Baltrušaitis, T., Johnson, M. & Shotton, J. *CONFIG: Controllable Neural Face Image Generation* in *European Conference on Computer Vision (ECCV)* (2020).

119. Deng, Y., Yang, J., Chen, D., Wen, F. & Tong, X. *Disentangled and Controllable Face Image Generation via 3D Imitative-Contrastive Learning* in *IEEE Computer Vision and Pattern Recognition* (2020).

120. Nguyen-Phuoc, T., Richardt, C., Mai, L., Yang, Y.-L. & Mitra, N. *BlockGAN: Learning 3D Object-aware Scene Representations from Unlabelled Images* in *Advances in Neural Information Processing Systems 33* (2020).

121. Shi, Y., Aggarwal, D. & Jain, A. *Lifting 2D StyleGAN for 3D-Aware Face Generation* in (2021), 6254.

122. Pan, X., Dai, B., Liu, Z., Loy, C. C. & Luo, P. *Do 2D GANs Know 3D Shape? Unsupervised 3D Shape Reconstruction from 2D Image GANs* in *International Conference on Learning Representations* (2021).

123. Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R. & Ng, R. *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis* in *ECCV* (2020).

124. Baatz, H., Granskog, J., Papas, M., Rousselle, F. & Novák, J. NeRF-Tex: Neural Reflectance Field Textures. *Computer Graphics Forum* **41**, 287 (2022).

125. Barron, J. T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R. & Srinivasan, P. P. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (2021).

126. Sara Fridovich-Keil and Alex Yu, Tancik, M., Chen, Q., Recht, B. & Kanazawa, A. *Plenoxels: Radiance Fields without Neural Networks* in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2022).

127. Müller, T., Evans, A., Schied, C. & Keller, A. Instant neural graphics primitives with a multiresolution hash encoding. *arXiv preprint arXiv:2201.05989* (2022).

128. Martin-Brualla, R., Radwan, N., Sajjadi, M. S. M., Barron, J. T., Dosovitskiy, A. & Duckworth, D. *NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections* 2020.

129. Guo, Y.-C., Kang, D., Bao, L., He, Y. & Zhang, S.-H. *NeRFReN: Neural Radiance Fields With Reflections* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), 18409.

130. Wu, L., Lee, J. Y., Bhattad, A., Wang, Y. & Forsyth, D. *DIVeR: Real-time and Accurate Neural Radiance Fields with Deterministic Integration for Volume Rendering* 2021.

131. Schwarz, K., Liao, Y., Niemeyer, M. & Geiger, A. *GRAF: Generative Radiance Fields for 3D-Aware Image Synthesis* in *Advances in Neural Information Processing Systems (NeurIPS)* (2020).

132. Chan, E. R., Monteiro, M., Kellnhofer, P., Wu, J. & Wetzstein, G. *pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis* in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2021), 5799.

133. Or-El, R., Luo, X., Shan, M., Shechtman, E., Park, J. J. & Kemelmacher-Shlizerman, I. *StyleSDF: High-Resolution 3D-Consistent Image and Geometry Generation* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), 13503.

134. Gu, J., Liu, L., Wang, P. & Theobalt, C. *StyleNeRF: A Style-based 3D Aware Generator for High-resolution Image Synthesis* in *International Conference on Learning Representations* (2022).

135. Skorokhodov, I., Tulyakov, S., Wang, Y. & Wonka, P. EpiGRAF: Rethinking training of 3D GANs. *arXiv preprint arXiv:2206.10535* (2022).

136. Niemeyer, M. & Geiger, A. *Giraffe: Representing scenes as compositional generative neural feature fields* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), 11453.

137. Schwarz, K., Sauer, A., Niemeyer, M., Liao, Y. & Geiger, A. Voxgraf: Fast 3d-aware image synthesis with sparse voxel grids. *arXiv preprint arXiv:2206.07695* (2022).

138. Shoshan, A., Bhonker, N., Kviatkovsky, I. & Medioni, G. *GAN-Control: Explicitly Controllable GANs* in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (2021), 14083.

139. Hu, Y., Wu, X., Yu, B., He, R. & Sun, Z. Pose-guided photorealistic face rotation. *CVPR* (2018).

140. Müller, T., Evans, A., Schied, C. & Keller, A. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Trans. Graph.* **41**, 102:1 (2022).

141. Sun, C., Sun, M. & Chen, H.-T. *Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), 5459.

142. Chen, A., Xu, Z., Geiger, A., Yu, J. & Su, H. *TensoRF: Tensorial Radiance Fields* in *European Conference on Computer Vision (ECCV)* (2022).

143. Wizadwongsa, S., Phongthawee, P., Yenphraphai, J. & Suwajanakorn, S. *NeX: Real-time View Synthesis with Neural Basis Expansion* in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2021).

144. Tucker, R. & Snavely, N. *Single-view View Synthesis with Multiplane Images* in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2020).

145. Yu, A., Li, R., Tancik, M., Li, H., Ng, R. & Kanazawa, A. *PlenOctrees for Real-time Rendering of Neural Radiance Fields* in *ICCV* (2021).

146. Garbin, S. J., Kowalski, M., Johnson, M., Shotton, J. & Valentin, J. FastNeRF: High-Fidelity Neural Rendering at 200FPS (2021).

147. Hedman, P., Srinivasan, P. P., Mildenhall, B., Barron, J. T. & Debevec, P. Baking Neural Radiance Fields for Real-Time View Synthesis. *ICCV* (2021).

148. Neff, T., Stadlbauer, P., Parger, M., Kurz, A., Mueller, J. H., Chaitanya, C. R. A., Kaplanyan, A. S. & Steinberger, M. DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. *Computer Graphics Forum* **40** (2021).

149. Hu, T., Liu, S., Chen, Y., Shen, T. & Jia, J. *EfficientNeRF Efficient Neural Radiance Fields* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), 12902.

150. Zhao, X., Ma, F., Güera, D., Ren, Z., Schwing, A. G. & Colburn, A. *Generative Multiplane Images: Making a 2D GAN 3D-Aware* in *Proc. ECCV* (2022).

151. Trevithick, A., Chan, M., Stengel, M., Chan, E. R., Liu, C., Yu, Z., Khamis, S., Chandraker, M., Ramamoorthi, R. & Nagano, K. *Real-Time Radiance Fields for Single-Image Portrait View Synthesis* in *ACM Transactions on Graphics (SIGGRAPH)* (2023).

152. Kwak, J.-g., Li, Y., Yoon, D., Kim, D., Han, D. & Ko, H. *Injecting 3D Perception of Controllable NeRF-GAN into StyleGAN for Editable Portrait Image Synthesis* in *European Conference on Computer Vision* (2022), 236.

153. Karras, T., Aittala, M., Laine, S., Härkönen, E., Hellsten, J., Lehtinen, J. & Aila, T. Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems* **34**, 852 (2021).

154. Karras, T., Laine, S. & Aila, T. *A Style-Based Generator Architecture for Generative Adversarial Networks* in (2019), 4396.

155. Choi, Y., Uh, Y., Yoo, J. & Ha, J.-W. *Stargan v2: Diverse image synthesis for multiple domains* in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020), 8188.

156. Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., *et al.* Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012* (2015).

157. Deng, Y., Yang, J., Xu, S., Chen, D., Jia, Y. & Tong, X. *Accurate 3D Face Reconstruction with Weakly-Supervised Learning: From Single Image to Image Set* in *IEEE Computer Vision and Pattern Recognition Workshops* (2019).

158. Deng, J., Guo, J., Niannan, X. & Zafeiriou, S. *ArcFace: Additive Angular Margin Loss for Deep Face Recognition* in *CVPR* (2019).

159. Roich, D., Mokady, R., Bermano, A. H. & Cohen-Or, D. Pivotal tuning for latent-based editing of real images. *ACM Transactions on Graphics (TOG)* **42**, 1 (2022).

160. Lin, C.-H., Gao, J., Tang, L., Takikawa, T., Zeng, X., Huang, X., Kreis, K., Fidler, S., Liu, M.-Y. & Lin, T.-Y. *Magic3D: High-Resolution Text-to-3D Content Creation* in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2023).

161. Wang, Z., Lu, C., Wang, Y., Bao, F., Li, C., Su, H. & Zhu, J. ProlificDreamer: High-Fidelity and Diverse Text-to-3D Generation with Variational Score Distillation. *arXiv preprint arXiv:2305.16213* (2023).

162. Liu, M., Xu, C., Jin, H., Chen, L., T, M. V., Xu, Z. & Su, H. *One-2-3-45: Any Single Image to 3D Mesh in 45 Seconds without Per-Shape Optimization* 2023.

163. Liu, Y., Lin, C., Zeng, Z., Long, X., Liu, L., Komura, T. & Wang, W. SyncDreamer: Learning to Generate Multiview-consistent Images from a Single-view Image. *arXiv preprint arXiv:2309.03453* (2023).

164. Qian, G., Mai, J., Hamdi, A., Ren, J., Siarohin, A., Li, B., Lee, H.-Y., Skorokhodov, I., Wonka, P., Tulyakov, S. & Ghanem, B. Magic123: One Image to High-Quality 3D Object Generation Using Both 2D and 3D Diffusion Priors. *arXiv preprint arXiv:2306.17843* (2023).

165. Long, X., Guo, Y.-C., Lin, C., Liu, Y., Dou, Z., Liu, L., Ma, Y., Zhang, S.-H., Habermann, M., Theobalt, C. & Wang, W. *Wonder3D: Single Image to 3D using Cross-Domain Diffusion* 2023.

166. Bautista, M. A., Guo, P., Abnar, S., Talbott, W., Toshev, A., Chen, Z., Dinh, L., Zhai, S., Goh, H., Ulbricht, D., Dehghan, A. & Susskind, J. GAUDI: A Neural Architect for Immersive 3D Scene Generation. *arXiv* (2022).

167. Chan, E. R., Lin, C. Z., Chan, M. A., Nagano, K., Pan, B., Mello, S. D., Gallo, O., Guibas, L., Tremblay, J., Khamis, S., Karras, T. & Wetzstein, G. *Efficient Geometry-aware 3D Generative Adversarial Networks* in *arXiv* (2021).

168. Ntavelis, E., Siarohin, A., Olszewski, K., Wang, C., Gool, L. V. & Tulyakov, S. *AutoDecoding Latent 3D Diffusion Models* 2023.

169. Mirzaei, A., Aumentado-Armstrong, T., Brubaker, M. A., Kelly, J., Levinshtein, A., Derpanis, K. G. & Gilitschenski, I. *Watch Your Steps: Local Image and Scene Editing by Text Instructions* in *arXiv* (2023).

170. Song, H., Choi, S., Do, H., Lee, C. & Kim, T. *Blending-NeRF: Text-Driven Localized Editing in Neural Radiance Fields* in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (2023), 14383.

171. Weder, S., Garcia-Hernando, G., Monszpart, Á., Pollefeys, M., Brostow, G., Firman, M. & Vicente, S. *Removing Objects From Neural Radiance Fields* in *CVPR* (2023).

172. Mirzaei, A., Aumentado-Armstrong, T., Derpanis, K. G., Kelly, J., Brubaker, M. A., Gilitschenski, I. & Levinshtein, A. *SPIn-NeRF: Multiview Segmentation and Perceptual Inpainting with Neural Radiance Fields* in *CVPR* (2023).

173. Yin, Y., Fu, Z., Yang, F. & Lin, G. *OR-NeRF: Object Removing from 3D Scenes Guided by Multiview Segmentation with Neural Radiance Fields* 2023.

174. Park, J., Kwon, G. & Ye, J. C. *ED-NeRF: Efficient Text-Guided Editing of 3D Scene using Latent Space NeRF* 2023.

175. Yu, L., Xiang, W. & Han, K. *Edit-DiffNeRF: Editing 3D Neural Radiance Fields using 2D Diffusion Model* 2023.

176. Ruiz, N., Li, Y., Jampani, V., Pritch, Y., Rubinstein, M. & Aberman, K. DreamBooth: Fine Tuning Text-to-image Diffusion Models for Subject-Driven Generation (2022).

177. Bartrum, E., Nguyen-Phuoc, T., Xie, C., Li, Z., Khan, N., Avetisyan, A., Lanman, D. & Xiao, L. *ReplaceAnything3D: Text-Guided 3D Scene Editing with Compositional Neural Radiance Fields* 2024.

178. Li, Y., Dou, Y., Shi, Y., Lei, Y., Chen, X., Zhang, Y., Zhou, P. & Ni, B. *FocalDreamer: Text-driven 3D Editing via Focal-fusion Assembly* 2023.

179. Shum, K. C., Kim, J., Hua, B.-S., Nguyen, D. T. & Yeung, S.-K. *Language-driven Object Fusion into Neural Radiance Fields with Pose-Conditioned Dataset Updates* 2023.

180. Brooks, T., Holynski, A. & Efros, A. A. *InstructPix2Pix: Learning to Follow Image Editing Instructions* in *CVPR* (2023).

181. Hertz, A., Mokady, R., Tenenbaum, J., Aberman, K., Pritch, Y. & Cohen-Or, D. Prompt-to-prompt image editing with cross attention control (2022).

182. Zhang, S., Yang, X., Feng, Y., Qin, C., Chen, C.-C., Yu, N., Chen, Z., Wang, H., Savarese, S., Ermon, S., Xiong, C. & Xu, R. HIVE: Harnessing Human Feedback for Instructional Visual Editing. *arXiv preprint arXiv:2303.09618* (2023).

183. Lugmayr, A., Danelljan, M., Romero, A., Yu, F., Timofte, R. & Van Gool, L. *RePaint: Inpainting Using Denoising Diffusion Probabilistic Models* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), 11461.

184. Avrahami, O., Lischinski, D. & Fried, O. *Blended Diffusion for Text-Driven Editing of Natural Images* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), 18208.

185. Patashnik, O., Garibi, D., Azuri, I., Averbuch-Elor, H. & Cohen-Or, D. *Localizing Object-level Shape Variations with Text-to-Image Diffusion Models* in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (2023).

186. Zhang, K., Mo, L., Chen, W., Sun, H. & Su, Y. *MagicBrush: A Manually Annotated Dataset for Instruction-Guided Image Editing* in *Advances in Neural Information Processing Systems* (2023).

187. Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Gontijo-Lopes, R., Ayan, B. K., Salimans, T., Ho, J., Fleet, D. J. & Norouzi, M. *Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding* in *Advances in Neural Information Processing Systems* (eds Oh, A. H., Agarwal, A., Belgrave, D. & Cho, K.) (2022).

188. Deitke, M., Schwenk, D., Salvador, J., Weihs, L., Michel, O., VanderBilt, E., Schmidt, L., Ehsani, K., Kembhavi, A. & Farhadi, A. *Objaverse: A Universe of Annotated 3D Objects* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2023), 13142.

189. Ranftl, R., Lasinger, K., Hafner, D., Schindler, K. & Koltun, V. Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-Shot Cross-Dataset Transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **44** (2022).

190. Schwarz, K., Liao, Y., Niemeyer, M. & Geiger, A. *GRAF: Generative Radiance Fields for 3D-Aware Image Synthesis* in *Advances in Neural Information Processing Systems (NeurIPS)* (2020).

191. Barron, J. T., Mildenhall, B., Verbin, D., Srinivasan, P. P. & Hedman, P. *Mip-nerf 360: Unbounded anti-aliased neural radiance fields* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), 5470.

192. Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G. & Sutskever, I. *Learning Transferable Visual Models From Natural Language Supervision* in *Proceedings of the 38th International Conference on Machine Learning* (eds Meila, M. & Zhang, T.) **139** (PMLR, 2021), 8748.

193. Gal, R., Alaluf, Y., Atzmon, Y., Patashnik, O., Bermano, A. H., Chechik, G. & Cohen-Or, D. *An Image is Worth One Word: Personalizing Text-to-Image Generation using Textual Inversion* 2022.

194. Kumari, N., Zhang, B., Zhang, R., Shechtman, E. & Zhu, J.-Y. Multi-Concept Customization of Text-to-Image Diffusion (2023).

195. Zhang, L., Rao, A. & Agrawala, M. *Adding Conditional Control to Text-to-Image Diffusion Models* 2023.

196. Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L. & Chen, W. *LoRA: Low-Rank Adaptation of Large Language Models* in *International Conference on Learning Representations* (2022).

197. Li, Y., Wang, H., Jin, Q., Hu, J., Chemerys, P., Fu, Y., Wang, Y., Tulyakov, S. & Ren, J. *SnapFusion: Text-to-Image Diffusion Model on Mobile Devices within Two Seconds* in *Thirty-seventh Conference on Neural Information Processing Systems* (2023).

198. Zhao, Y., Xu, Y., Xiao, Z. & Hou, T. *MobileDiffusion: Subsecond Text-to-Image Generation on Mobile Devices* 2023.

199. Kerbl, B., Kopanas, G., Leimkühler, T. & Drettakis, G. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* **42** (2023).

200. Chen, Y., Chen, Z., Zhang, C., Wang, F., Yang, X., Wang, Y., Cai, Z., Yang, L., Liu, H. & Lin, G. *GaussianEditor: Swift and Controllable 3D Editing with Gaussian Splatting* 2023.

201. Blattmann, A., Rombach, R., Ling, H., Dockhorn, T., Kim, S. W., Fidler, S. & Kreis, K. *Align your Latents: High-Resolution Video Synthesis with Latent Diffusion Models* in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2023).

202. Blattmann, A., Dockhorn, T., Kulal, S., Mendelevitch, D., Kilian, M., Lorenz, D., Levi, Y., English, Z., Voleti, V., Letts, A., Jampani, V. & Rombach, R. *Stable Video Diffusion: Scaling Latent Video Diffusion Models to Large Datasets* 2023.

203. Bar-Tal, O., Chefer, H., Tov, O., Herrmann, C., Paiss, R., Zada, S., Ephrat, A., Hur, J., Liu, G., Raj, A., Li, Y., Rubinstein, M., Michaeli, T., Wang, O., Sun, D., Dekel, T. & Mosseri, I. *Lumiere: A Space-Time Diffusion Model for Video Generation* 2024.

204. Peruzzo, E., Goel, V., Xu, D., Xu, X., Jiang, Y., Wang, Z., Shi, H. & Sebe, N. *VASE: Object-Centric Appearance and Shape Manipulation of Real Videos* 2024.

205. Kahatapitiya, K., Karjauv, A., Abati, D., Porikli, F., Asano, Y. M. & Habibian, A. *Object-Centric Diffusion for Efficient Video Editing* 2024.

206. Bahmani, S., Skorokhodov, I., Rong, V., Wetzstein, G., Guibas, L., Wonka, P., Tulyakov, S., Park, J. J., Tagliasacchi, A. & Lindell, D. B. 4D-fy: Text-to-4D Generation Using Hybrid Score Distillation Sampling. *arXiv* (2023).

207. Shao, R., Sun, J., Peng, C., Zheng, Z., Zhou, B., Zhang, H. & Liu, Y. Control4D: Efficient 4D Portrait Editing with Text (2023).

208. Kynkäänniemi, T., Karras, T., Laine, S., Lehtinen, J. & Aila, T. Improved Precision and Recall Metric for Assessing Generative Models. *CoRR* **abs/1904.06991** (2019).

209. Richardson, E., Alaluf, Y., Patashnik, O., Nitzan, Y., Azar, Y., Shapiro, S. & Cohen-Or, D. *Encoding in Style: a StyleGAN Encoder for Image-to-Image Translation* in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021).

210. *EG3D Inversion Projector* `https://github.com/oneThousand1000/EG3D-projector`.