DART-IDE: A Latent Diffusion Model of Articulated and Textured 3D Shapes with Interpretable Direction Editing

Patrikas Vanagas

Master of Science
Artificial Intelligence
School of Informatics
University of Edinburgh
2024

Abstract

Recent advances in generative AI have revolutionised the creation of 2D content but have lagged in generating 3D models. In this thesis, we introduce Diffusing Articulated Renderings of Textured Meshes with Interpretable Direction Editing, a novel framework that integrates diffusion models with the recently proposed Single-View Articulated Object Reconstruction model for generating articulated and textured 3D meshes. DART-IDE independently generates and controls shape, articulation, and texture, addressing current limitations in 3D mesh generation. We also make contributions to latent space disentanglement, being the first to enable interpretable direction editing specifically in *class-conditional* diffusion models, and apply this to DART-IDE, enabling predictable fine-grained manipulation of the pose of generated meshes. We also propose a new metric, the Articulated Mesh Pairwise Chamfer Distance, to evaluate the quality and diversity of generated 3D meshes. Experiments demonstrate that DART-IDE generates high-quality, diverse 3D meshes with meaningful and predictable edits, offering significant advancements over existing models. The proposed methods show promise in enhancing 3D generative capabilities, opening new avenues for further research.

Research Ethics Approval

This project was planned according to the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Patrikas Vanagas*)

Acknowledgements

I thank my supervisor, Dr Oisin Mac Aodha, for being very responsive and steering me in the right direction throughout this project, and Mehmet Aygün, who kept the SAOR code in accessible shape. I am grateful to my family, who have been a constant source of support and encouragement. I also thank my library companions and friends, both near and far, for their companionship and for providing the rare, but much-needed moments of levity. This work has used the resources provided by the Cirrus UK National Tier-2 HPC Service at Edinburgh Parallel Computing Centre [1] funded by the University of Edinburgh and EPSRC (EP/P020267/1); I extend my sincere thanks to the Cirrus support team for their swift and efficient assistance.

[1] www.cirrus.ac.uk

Table of Contents

Acronyms & Initialisms

DART-IDE **D**iffusing **A**rticulated **R**enderings of **T**extured Meshes with **I**nterpretable **D**irection **E**diting

CNN Convolutional Neural Network

DDPM Denoising Diffusion Probabilistic Model

DDIM Denoising Diffusion Implicit Model

SAOR Single-View Articulated Object Reconstruction

VAE Variational Autoencoder

GAN Generative Adversarial Network

SVR Single-View 3D Object Reconstruction

SSL Self-Supervised Learning

MLP Multi-Layer Perceptron

ELBO Evidence Lower Bound

KLD Kullback–Leibler divergence

LSGM Latent Space Generative Model

CFG Classifier-Free Diffusion Guidance

PCA Principal Component Analysis

AMPCD Average Minimum Pairwise Chamfer Distance

FID Fréchet Inception Distance

NeRF Neural Radiance Field

Chapter 1

Introduction

1.1 Motivation

Recent advances in generative AI for images and videos, particularly with models such as DALL-E [94], StyleGAN [48], and the recent SORA [76], have enabled unprecedented levels of complexity in content generation [100]. However, generative AI has lagged in the generation of 3D content, despite its vast potential for applications in gaming, VR/AR, architecture, e-commerce, healthcare, manufacturing, education, and the arts, where it can revolutionise design, visualisation, and interactive experiences by enabling the rapid creation of detailed and customisable 3D models for everything from virtual environments and product prototypes to medical simulations and digital art ([62], [5]). Progress in 3D AI is hindered by the complexity of 3D representations, which require handling higher-dimensional data compared to 2D images, the scarcity of realistic large-scale 3D datasets for training, the challenges in ensuring multiview consistency in generated models, and the computational intensity of rendering and optimising 3D content [67]. Additional challenges include the difficulty in generating compact and topologically accurate parametrisations, the inability to produce detailed textures and material properties, and the challenges in ensuring consistency and precision when generating 3D content based on conditional inputs such as labels, text, or images [71], making it difficult to achieve the same level of quality and diversity in 3D generation as seen in 2D content generation. In part because of this, few metrics have been developed for evaluating generated 3D content encoded in *meshes*, which is the parameterisation allowing the most down-the-line manipulation, and no work has been done on *interpretable controllability* of such generated objects, allowing for predictable manipulation during generation, where one a priori knows the controlled attribute.

Another related field, which has seen more progress in the last few years, is that of image-to-3D models ([72], [73]), where the 3D object is inferred from a single image. At the same time, progress in the generation of *articulated* 3D objects from images has been constrained by the need for extensive supervision, such as multiview images, keypoint-derived camera poses, or predefined shape assumptions, limiting the scalability and flexibility of these methods [125]. One method that has recently successfully tackled this task by removing some means of supervision is Single-View Articulated Object Reconstruction (SAOR) [4], the efficient scheme of which allows for quick and efficient inference of 3D textured and articulated animal meshes learnt from animal image collections via a self-supervised arrangement. In addition to being fast in inference and providing results of great quality, SAOR interests us because it contains *intermediate latent representations* which separately parametrise the shape, articulation and texture of the resultant 3D object. In light of this, one is tempted to use the recent advances in powerful generative models to learn the distribution of these latents and generate inputs to further modules of SAOR, thus turning it from a model which only works with image conditioning into a fully generative model, where shape, articulation, and texture generation is done independently.

In particular, Denoising Diffusion Probabilistic Models (DDPMs) [40], having emerged as a powerful generative paradigm, offer a promising avenue for modelling the complex distributions of the intermediate latents in SAOR, making it feasible to create a model capable of generating high-quality, diverse 3D articulated meshes with independent control over shape, articulation, and texture. This approach not only addresses the limitations of current 3D generative models, which struggle with detailed and conditional 3D content generation, but also opens new possibilities for **(i)** making the generative model class-conditionable, **(ii)** developing fine-grained, interpretable manipulation of 3D objects using the latest advancements in disentanglement of DDPMs, **(iii)** developing 3D mesh evaluation metrics having a ground truth distribution of original shapes.

1.2 Contributions

Developing our approach, which we coin **D**iffusing **A**rticulated **R**enderings of **T**extured Meshes with **I**nterpretable **D**irection **E**diting (DART-IDE), we make these contributions:

1. **A Novel Framework for Generating 3D Meshes:** We introduce DART-IDE, a novel framework that extends the capabilities of SAOR by integrating DDPMs for

the generative modelling of articulated and textured 3D meshes. This framework allows for independent generation and control of shape, articulation, and texture, addressing the limitations of current 3D generative models in producing high-quality, diverse, and conditionable 3D content.

2. **Interpretable Direction Discovery in Class-Conditional Diffusion Models**: We extend existing diffusion model disentanglement techniques to enable interpretable direction discovery within the latent space of *class-conditional* models, and apply it to DART-IDE. This allows interpretable manipulation of generated meshes *across different classes*, e.g., adjusting specific attributes in the pose.

3. **Development of a New Evaluation Metric:** We propose the Average Minimum Pairwise Chamfer Distance (AMPCD), a novel metric to evaluate generated 3D meshes. This metric specifically addresses the challenges of assessing generated articulated objects in terms of *both* quality and diversity, providing a more robust and meaningful evaluation compared to existing methods.

4. **A Novel Animal Image Dataset:** In the development of DART-IDE, we curate a dataset of 14,352 animal images from 16 classes, on which we expect most image-to-3D models to perform very well. From this dataset, we construct the dataset of SAOR latents to train DART-IDE.

To the best of our knowledge, DART-IDE is the first generative 3D mesh model to have mechanisms to independently generate and manipulate the shape, articulation, and texture of a mesh. Furthermore, because of our contributions to diffusion model disentanglement, it is the first to do so in a predictable manner.

1.3 Thesis Outline

This thesis is organised as follows: Chapter 2 provides a foundational background on computer vision, 3D object reconstruction, and latent space generative models, focussing on the SAOR model and its integration into DART-IDE. Chapter 3 reviews the relevant diffusion model theory and advancements we use, setting the stage for DART-IDE's architecture. Chapter 4 details the methodology, including the generation scheme and model architectures, dataset preparation, evaluation metrics, and latent space disentanglement. Chapter 5 presents and analyses our results. Finally, Chapter 6 concludes with a summary and future directions.

Chapter 2

Foundations

In this chapter, our aim is to provide the reader with the necessary foundations of the methodologies upon which we set out the project. First, we summarise recent progress in computer vision, especially based on deep learning. Next, we analyse recent work mapping images to 3D meshes, especially focussing on SAOR [4], the model that forms the backbone of our project. Finally, we discuss the technique of latent space generative models, which we merge with SAOR to create our model, DART-IDE.

2.1 Primer on Computer Vision

Computer vision seeks to enable computers to interpret visual data akin to human perception [110], having initially grown alongside computational human vision theory [81]. Key historical advances have included edge detection, crucial for scene understanding [21], image pyramids for blending and coarse-to-fine correspondence ([43], [14], [92], [3]), and global optimisation techniques like graph cuts for dense stereo correspondence [11]. Feature-based recognition, using models like constellation and pictorial structures, also greatly enhanced object recognition ([27], [26]).

However, the invention that may have achieved the most practical modern advances is the Convolutional Neural Network (CNN) [58], excelling due to translational invariance achieved through convolutional layers that ensure shift-invariant feature detection, and the advent of deep learning ([60], [56]), enabling to overcome human-level performance for some detection and classification tasks ([36], [19]). For 3D computer vision, deep learning has also been employed to enhance tasks such as object detection [135], pose estimation [108], and semantic segmentation [119], significantly improving the accuracy and efficiency of these applications [101].

Contrary to the techniques mentioned above, *generative* models aim to learn the underlying data distribution $p(x)$ and generate new samples $x \sim p(x)$. These models include Variational Autoencoders (VAEs) [51], Generative Adversarial Networks (GANs) [31], Normalizing Flows [96], Energy-Based Models [59], Autoregressive Models [114] and Diffusion Models [103], each employing distinct methods for learning and sampling from $p(x)$, thus providing powerful tools for a variety of applications such as image synthesis, super-resolution, and data augmentation [10]. We discuss models relevant to our work in more detail in sections Subsect. 2.3 and Chapter 3. Opposite to the powerful breakthroughs in audio [12], image [66] and text [25], results in 3D object generation have been rather more humble. Notable results include volumetric pixel (*voxel*) generation using VAEs [13] and GANs [124]. Later, GANs have been extended to create collections of discrete points in 3D space (*point clouds*) by making the generation process hierarchical [63] and by discrete patch generation [122]. Furthermore, works parametrising the object by a collection of vertices, edges and faces (a *mesh*) have been proposed using a double VAE structure [28] and geometry-aware 3D GANs [17].

Non-traditional representations of 3D objects have been gaining traction too. Neural Radiance Fields (NeRFs) [83], which, using a neural network, parametrise a 3D scene by representing it as a continuous volumetric function that maps spatial coordinates and viewing directions to colour and density values, have enabled using large pre-trained image diffusion models, described in Chapter 3, for generating 3D objects from text prompts. Notably, Poole et al. [90] have set a new standard by combining NeRFs with Score Distillation Sampling, effectively utilizing gradients from a pre-trained 2D diffusion model to fine-tune NeRF parameters. This method allows the generation of 3D objects from text prompts without relying on large labelled datasets, marking a significant breakthrough. Subsequent research, such as Wang et al. [121], has further refined this process through variational score distillation, while Liu et al. [74] improved efficiency by synthesizing views from minimal input images using geometric priors. Gaussian Splatting [49], which uses an explicit scene representation with 3D Gaussians that are projected onto the image plane for faster rendering rather than a continuous parametrisation, have demonstrated impressive results; for example, Zhang et al. [133] demonstrate that progressive densification in Gaussian Splatting allows faster optimisation compared to NeRFs, while Tang et al. [111] demonstrate that their Large Multi-View Gaussian Model surpasses previous methods in both resolution and efficiency, establishing it as a leading framework for high-fidelity 3D content generation

from text or single-view images.

However, it is by mastering the mesh generation modus operandi by which we reap the most advantages; by this parametrisation, we enable the object to be highly optimized for rendering pipelines, directly define the surface of an object, which is essential for tasks that require surface detail, and make it easy to manipulate and edit. To the best of our knowledge, there has been no work that explicitly generates mesh-based representations that simultaneously model the joint texture distribution using techniques like UV mapping [106] or similar; furthermore, no mesh-based generator enables on-the-fly editing with interpretable generation directions. Among the few approaches addressing articulation in object generation, the NAP method [61] stands out. This framework generates articulated 3D models using a graph-attention denoising network that captures the relationships between part geometry and the motion constraints of joints. Despite its innovative approach, the articulation graphs it produces are relatively basic, especially when compared to the complexity required for models like animals, and it does not incorporate texture learning.

2.2 3D Object Reconstruction

Single-View 3D Object Reconstruction (SVR) in computer vision refers to the process of creating a three-dimensional digital model of an object from two-dimensional images or other data inputs such as point clouds or depth maps, and the first work that attempts to do this [97] is usually referred to as the groundwork for future vision work. Despite modern approaches being rather successful in the quality of the outputs, they require additional supervision besides images, such as **(i)** multi-view images of the same object [129], **(ii)** camera poses from keypoints [75], **(iii)** object silhouettes [42], **(iv)** making some a priori assumptions about the shape, such as on the object's template shape or symmetry ([29], [112], [84]), or **(v)** using manually defined 3D skeleton supervision ([126], [127]). Therefore, the state of the current field may be summarised as making strides to remove as much additional supervision as possible while achieving the highest quality of the generated shapes. Due to this, *Self-Supervised Learning (SSL)*, where a pretext task is used to produce intermediate representations to be used later, all while using unannotated data [6], is becoming a more and more promising direction.

SAOR [4] is an SSL method which focuses on textured and articulated animal reconstruction from 2D images; the task is especially challenging due to animals having highly deformable bodies. The model outperforms other existing methods that do not

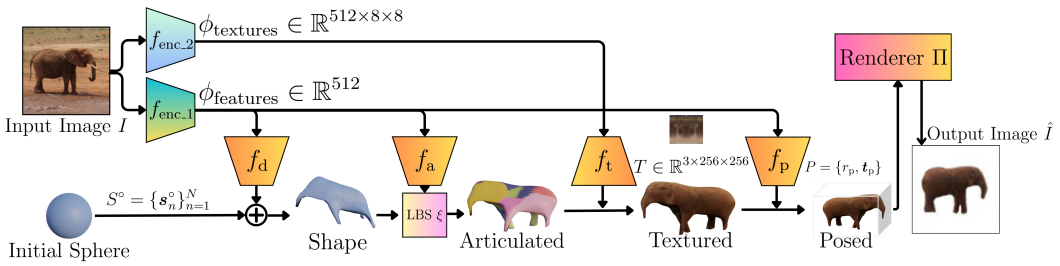

Figure 2.1: Overview of the SAOR model architecture and forward propagation (uses elements from Fig. 1 from [4]). The model predicts shape deformation, articulation, camera viewpoint, and texture from a single input image using separate encoders and modules: global latents ϕ_{texture} and ϕ_{features} are extracted by $f_{\text{enc\_2}}$ and $f_{\text{enc\_1}}$; these get decoded by f_{d}, f_{a}, f_{t} and f_{p}, to generate the final output image $\hat{I}$ so that it is depicted from the same viewpoint as the initial image.

use explicit 3D supervision on the Percentage of Correct Keypoints metric for the CUB dataset [117], and allows for quick inference on limited GPU resources, essential for our application. Below, we investigate the components of the computation graph for SAOR, which is crucial for understanding the arrangement of the generative version of the model. We note that the version of SAOR we use is slightly modified compared to the one described in the paper - ours uses two separate global encoders, whereas the original paper used just one. We show the inference stage of SAOR in Fig. 2.1. The forward propagation begins with the extraction of global image representations, $\phi_{\text{features}} \in \mathbb{R}^{512}$ and $\phi_{\text{texture}} \in \mathbb{R}^{512\times8\times8}$, from an input image $I \in \mathbb{R}^{3\times128\times128}$ using two separate ResNet-18 [35] encoders. These global variables are used to predict several key components: shape deformation, articulation, camera viewpoint, and texture.

1. Shape Prediction: The initial shape is modelled as a sphere mesh $S^{\circ} = \{\boldsymbol{s}_n^{\circ}\}_{n=1}^{N}$ where each $\boldsymbol{s}_n^{\circ}$ represents the 3D coordinates of a vertex. Sphere initialized mesh being held in a `PyTorch3D` object (which holds the 3D coordinates of vertices, triplets specifying the faces and the texture image), the deformed shape S' is predicted using a field modelled by coordinate-based Multi-Layer Perceptrons (MLPs), and each vertex gets transformed as

$$\boldsymbol{s}_i' = \boldsymbol{s}_i^{\circ} + f_d(\boldsymbol{s}_i^{\circ}, \phi_{\text{im}}). \tag{2.1}$$

Here, f_d outputs the displacement vector for the initial points s_i°. Given the bilateral symmetry of most natural objects, only the vertices on the positive side of the xy-plane

are deformed, and the deformation is reflected for the vertices on the negative side.

2. Articulation Prediction: Articulation is applied to the deformed shape S' using a skeleton-free linear blend skinning [55] method. This allows the model to apply realistic deformations to the object parts without relying on predefined skeletal structures. The final articulated shape S is computed as:

$$S = \xi(S', A), \tag{2.2}$$

where ξ is the LBS operation, and A consists of the part assignment matrix W, found by an additional MLP acting on vertices and ϕ_{features}, which signifies the probability of each vertex belonging to a part and is later softmaxed, and transformation parameters $\pi = \{\boldsymbol{z}_k, \boldsymbol{r}_k, \boldsymbol{t}_k\}_{k=1}^{K}$, which include scale $\boldsymbol{z}_k \in \mathbb{R}^3$, rotation $r_k \in \mathbb{R}^{3\times3}$, and translation $\boldsymbol{t}_k \in \mathbb{R}^3$ for each part $k = 12$, all found by different MLPs acting on ϕ_{features}. The vertex positions are articulated based on

$$\boldsymbol{s}_i = \sum_{k=1}^{K} W_{i,k} \boldsymbol{z_k} \odot \left(r_k(\boldsymbol{s}'_i - \boldsymbol{c}_k) + \boldsymbol{t}_k \right), \tag{2.3}$$

where $\boldsymbol{c}_k$ is the center of part k:

$$\boldsymbol{c}_k = \frac{\sum_{i=1}^{N} \boldsymbol{s}'_i \cdot W_{i,k}}{\sum_{i=1}^{N} W_{i,k}}. \tag{2.4}$$

This LBS mechanism blends transformations across parts using the weights $W_{i,k}$, producing smooth and natural articulations that are critical for reconstructing the highly deformable bodies of animals.

3. Texture Prediction: The texture image T of the object is generated using the ϕ_{texture} latent and the faces of the deformed shape. ϕ_{texture} is first upscaled using a 7-layer CNN decoder f_t from [86]:

$$T = f_t(\phi_{\text{texture}}), \tag{2.5}$$

where $T \in \mathbb{R}^{3\times256\times256}$, and is then horizontally mirrored to $T \in \mathbb{R}^{3\times512\times256}$ to satisfy the bilateral symmetry. This image is directly mapped onto the faces using the 2D UV coordinate system, where each of the 3 vertices of each face is assigned a UV coordinate. Additionally, after this step, the coordinates of all vertices get multiplied by

a scalar value, predicted from a separate MLP taking in ϕ_{features}, to scale up the size of the shape.

4. Camera Pose Prediction: The camera pose P, parameterized by a rotation matrix $r_p \in \mathbb{R}^{3\times3}$ and translation $\boldsymbol{t}_p \in \mathbb{R}^3$, is predicted by separate MLP regressors from ϕ_{features}. 4 possible P's are guessed, and the most likely one is picked by a separate probability MLP. The mesh vertices are then rotated and translated by the found configuration.

5. Rendering: Finally, the predicted shape S, texture T, and camera pose P are used to render the final image $\hat{I}$:

$$\hat{I} = \Pi(S, T, P) \tag{2.6}$$

where Π denotes the differentiable rendering operation. The forward propagation process, thus, allows the SAOR model to predict and reconstruct the 3D shape, texture, and viewpoint of the object from a single input image.

The model is trained with an end-to-end SSL analysis-by-synthesis framework, that is, minimizing the discrepancy between the rendered reconstruction image $\hat{I}$ and the input image I from Eq. 2.6, using LSUN [131] horse images for warming up and a custom dataset of 90k images of 101 animal classes from iNaturalist [44]. The loss contains linear combinations of terms which penalize **(i) Appearance Differences** The L2 RGB pixel value distance between the two images, and difference between values of activations of a pre-trained VGG-16 CNN [102] which are observed having passed the two images through **(ii) Mask Differences** The L2 loss of the segmentation mask of the original image predicted using an off-the-shelf Segment Anything Model [53] and one from the rendered image (which does not include the background) **(iii) Depth Differences** The L2 loss of the depth predicted from the original image using an off-the-shelf MiDaS model [9] compared to the one from the rendered image **(iv) Multi-View Inconsistency** A swap loss compares the original image to a synthetic one created by swapping shape encodings between instances, maintaining other attributes. This ensures consistency across views and prevents degenerate (e.g. flat) 3D shapes **(v) Roughness and Part Assignment Variation** The regularization loss, encouraging equal-sized parts and smooth transitions between them, penalizing sharp edges and irregular part assignments, to ensure the reconstructed 3D shape is both realistic and coherent.

Therefore, using off-the-shelf solutions for depth and segmentation predictions *only* during training and only assuming unarticulated animal symmetry, SAOR provides an efficient way to output articulated meshes with textures of great quality during inference time, well-fitting for our task - inference on a single Nvidia RTX3060 GPU

takes around 500 milliseconds, while on an Nvidia V100 GPU it takes milliseconds. However, we acknowledge concurrent SVR work to SAOR which could have been used to construct a similar model to DART-IDE. Farm3D [46] solely uses a pre-trained image generation diffusion model for its synthetic training data, and incorporates it in a way where the generator is asked to provide the views of an object from different angles and illumination for virtual multi-view supervision. 3DFauna [69] uses a bank of possible shapes, coined the Semantic Bank of Skinned Models, which gets optimized during training to provide approximations for possible shapes to be inferred. Both models seem to show fewer degeneration cases than SAOR, but at the time of writing, the code for the models is not yet publicly available, and the inference times stated in the respective papers are in the range of a few seconds.

2.3 Latent Space Generative Models

Latent Space Generative Models (LSGMs) are generative models in which one learns to map the input data to a lower-dimensional "latent space" and then generates new data by sampling from this space, perhaps the best-known example being VAEs, introduced in Kingma and Welling [51]. Instead of learning to map input data $\mathbf{x}$ to a latent representation $\mathbf{z}$ through an encoder as in simple autoencoders and then map $\mathbf{z}$ back to $\mathbf{x}'$ through a decoder, VAEs impose a probabilistic structure by learning a distribution of latent $\mathbf{z}$, that is, $q_\phi(\mathbf{z}|\mathbf{x})$, which is typically set to be a Gaussian, parametrised by its mean(s) and standard deviation(s). The structure used is almost identical to a simple autoencoder, but the bottleneck neurons are set to be *stochastic*, that is, provide samples through the mean and standard deviation obtained through the encoder. Due to the intractable posterior, training is performed by minimising the Evidence Lower Bound (ELBO)

$$\mathcal{L} = \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p(\mathbf{z})}\right]}_{\text{KL Divergence}}, \tag{2.7}$$

where $p(\mathbf{z})$ is a prior, usually taken as a standard Gaussian $\mathcal{N}(0, I)$ [52]. The first term corresponds to how well the VAE can reconstruct the input data, and in the case of images, could be the MSE loss between the ground truth and the recovered image. The second term is the Kullback–Leibler divergence (KLD) between the approximate posterior and the prior, which acts as a regularizer, encouraging the learned distribution

to follow a form close to the prior. Training this type of model allows generating new data by sampling $\mathbf{z}$ from the prior $p(\mathbf{z})$ and passing it through the decoder $p_\theta(\mathbf{x}|\mathbf{z})$. This has allowed to generate high quality images ([95], [115]), audio ([15], [65]), point clouds ([1], [132]) and simple meshes [28].

However, modelling and then generating from latent distributions much more complicated than previous examples requires more complex models. Diffusion models, which have shown impressive results in a variety of modalities and the theory of which we discuss in the next chapter, have been proposed to be used as an LSGM by Vahdat et al. [113]. Here, one trains an autoencoder and then trains a diffusion model to recreate the distribution of intermediate latents. For generation, the generated latents are fed to the original decoder. For instance, [7] encode object point clouds into a latent space of $\mathbb{R}^{256\times 1}$, where a latent diffusion model is employed to learn the distribution of these latents and generate realistic robotic grasps. Similarly, Pinaya et al. [89] train a 3D convolutional autoencoder on MRI brain scans with bottleneck tensors of dimension $20 \times 20 \times 28$, developing a diffusion model conditioned on factors such as age, sex, and neuroanatomy to create a synthetic dataset of 100,000 scans. Additionally, Li et al. [68] adopt a very similar framework to generate realistic DNA sequences.

Chapter 3

Diffusion Models

Diffusion models have recently come to the fore as a powerful approach in machine learning, particularly for generating high-quality samples by modelling the gradual noise reduction process [130]. In this chapter, we review the techniques used in DART-IDE. Specifically, we discuss the theory behind DDPMs and then explore approaches that make the model faster to train, improve inference efficiency, enable conditional generation, and disentangle its latent space. We note that we describe the architectural details of DART-IDE in Chapter 4, as ours has improvements over the neural networks used in historical papers.

3.1 Denoising Diffusion Probabilistic Models

Diffusion models are a type of generative model inspired by physical nonequilibrium thermodynamics, which defines a Markov chain, where the probability of each transition depends only on the current state, not on the sequence of events that preceded it, of *diffusion steps*, where Gaussian noise is added to data, and afterwards a reverse process is learnt to generate new examples from the distribution. Several types of similar models have been proposed, namely Diffusion Probabilistic Models [103], which explicitly model the forward and reverse diffusion processes, Noise-Conditioned Score Networks [105], which directly estimate the gradient of the data distribution conditioned on the noise, and DDPMs [40], which iteratively denoise data through a learnt reverse process that gradually removes noise. We focus on the latter, as it is DDPMs that have shown the most groundbreaking results in most generational domains, from image synthesis [23] and image segmentation [2] to molecule generation [41]. Generally, the great results that have been achieved can be explained by the fact that DDPMs are less susceptible

to mode collapse and non-convergence problems than other models due to the highly efficient maximisation of likelihood and the gradual nature of denoising [8], allowing one to efficiently construct very large models trained on large amounts of data.
Introduced in Ho et al. [40], DDPMs require defining a *forward diffusion process* using samples from the data distribution $\mathbf{x}_0 \sim q(\mathbf{x}_0)$. The forward process $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ is defined as adding isotropic Gaussian noise to the data sample at times $t \in T$ to produce noisy samples $\mathbf{x}_1, ..., \mathbf{x}_T$, according to a variance schedule $\{\beta_t \in (0,1)\}_{t=1}^{T}$, which specifies the step size. Therefore, the forward process is

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}), \tag{3.1}$$

and

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \tag{3.2}$$

where $\mathbf{x}_T$ is isotropic Gaussian noise too. If one lets $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$, it can be proven [40] that the t-th sample $\mathbf{x}_t$ is

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}). \tag{3.3}$$

The goal of DDPMs is to successfully learn the reverse diffusion process $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, to start from Gaussian noise and denoise it and generate new samples from the data distribution, all while using a neural network with parameters θ. We show both forward and backward processes in Fig. 3.1. The reverse distribution is Gaussian as well [79], therefore can be be parametrised by a mean $\mu_\theta(\mathbf{x}_t, t)$ and variance $\Sigma_\theta(\mathbf{x}_t, t)$, both dependent on the timestep, so that

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)) \tag{3.4}$$

and

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t). \tag{3.5}$$

Reparametrising the equations to make the neural network (which is usually a CNN U-Net [98], outputs of which are of the same dimensionality as the inputs) ε_θ to *predict the noise added* at each timestep at time t, we can obtain the mean as

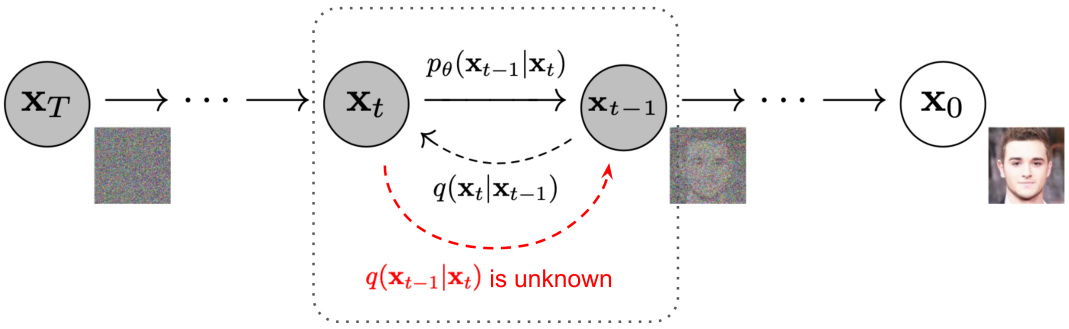

Figure 3.1: The forward and backward process Markov chain in a DDPM. Fig. 2 from Ho et al. [40], with additions from Weng [123].

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t)\right), \tag{3.6}$$

and then obtain $\mathbf{x}_{t-1}$ as

$$\text{Thus } \mathbf{x}_{t-1} = \mathcal{N}(\mathbf{x}_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t)\right), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)). \tag{3.7}$$

Using ELBO similarly to Eq. 2.7 to minimize the negative log-likelihood and obtain the ground truth sample $\mathbf{x}_0$ from the true distribution, we aim to minimize the KLD between two Gaussians [18]. The final optimisation objective turns out [40] to be taking a random ground truth sample $\mathbf{x}_0$ and, uniformly, a timestep index t, and then sampling an isotropic Gaussian noise of dimensionality of the data, adding it to the sample at that noise level in accordance with Eq. 3.3, and training the neural network to predict what noise had been added based on t and the noised $\mathbf{x}_t$:

$$\mathcal{L} = \mathbb{E}_{t\sim\mathcal{U}(1,T),\mathbf{x}_0\sim q(\mathbf{x}_0),\varepsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon}, t)\|^2\right]. \tag{3.8}$$

We show the DDPM training algorithm in Alg. 1. Having trained ε_θ well enough, one can generate new data points using Eq. 3.7. To accomplish this, one samples an isotropic Gaussian $\mathbf{x}_T$ and then using ε_θ with Eq. 3.7 arrives at a generated $\mathbf{x}_0$. However, besides this, to undeterministically arrive at different $\mathbf{x}_0$ at each generation attempt, at each t, one must sample an additional isotropic Gaussian ε, which is then added according to (an optionally different) noise schedule σ, which is usually set to β_t [79]. The DDPM sampling algorithm is shown in Alg. 2.

Algorithm 1 DDPM Training [40]

repeat
  $t \sim \text{Uniform}(\{1, \dots, T\})$
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{x}_t \leftarrow \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon}$
  Take gradient descent step on $\nabla_\theta \left\| \boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t) \right\|^2$
until converged

Algorithm 2 DDPM Sampling [40]

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for all t **from** T **to** 1 **do**
  $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\boldsymbol{\mu} \leftarrow \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t) \right)$
  $\mathbf{x}_{t-1} \leftarrow \boldsymbol{\mu} + \sigma_t \boldsymbol{\varepsilon}$
end for
return $\mathbf{x}_0$

3.2 Other Techniques Employed

Since the inception of DDPMs, many techniques have been added to make this class of models more adaptable and efficient. Here, we first review methods that we have used to efficiently condition the model on the class label of the generated data point. Afterwards, we look at means that allow many fewer timesteps than in DDPMs while still retaining the high quality of generated samples and allow for quicker training. Finally, we look at some approaches to disentangle the latent space of diffusion models.

3.2.1 Conditional Generation

In this work, we condition our diffusion models on class labels. With DDPMs, Dhariwal and Nichol [23] were the first to incorporate class information by taking an unconditional diffusion model and, during sampling, *injecting* the gradients of a classifier model $f_\phi(y|\mathbf{x}_t, t)$ that had been trained to separate the classes of noised images. Due to the fact that for a trained denoiser $\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t) = -\frac{1}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t)$, the new prediction of denoising $\bar{\varepsilon}_\theta$ conditioned on class y becomes

$$\bar{\boldsymbol{\varepsilon}}_\theta(\mathbf{x}_t, t) = \boldsymbol{\varepsilon}_\theta(x_t, t) - \sqrt{1-\bar{\alpha}_t}\, w \nabla_{\mathbf{x}_t} \log f_\phi(y|\mathbf{x}_t), \tag{3.9}$$

and in each iteration of Alg. 2, the obtained mean would become

$$\mu = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{x}_t}}\varepsilon_\theta(\mathbf{x}_t, t, y)) + \omega\sigma_t \nabla_{\mathbf{x}_t} \log p\phi(c|\mathbf{x}_t), \tag{3.10}$$

where ω controls the strength of the guidance, and larger w would be expected to result in better quality, but less diversity in the samples [33]. However, the downside of the method is that one cannot use a pre-trained classifier, and it may just be interpreted as an adversarial attack.

A more traditional conditioning method requiring twice as many diffusion steps, introduced by Ho and Salimans [39], is referred to as Classifier-Free Diffusion Guidance (CFG), and incorporates the output of *both* a conditional and unconditional diffusion model pass. The conditional diffusion model $p_\theta(\mathbf{x}|y)$ is trained on datapoints paired with labels c with a model $\varepsilon(\mathbf{x}_t, t, c)$, and from time to time during, the conditional information gets *dropped out*, with probability p_{uncond}, by setting $c \leftarrow \varnothing$. By Bayes rule, the full prediction $\tilde{p}_\theta$ becomes

$$\tilde{p}_\theta(\mathbf{x_t}|c) \propto p_\theta(\mathbf{x}_t|c) p_\theta(c|\mathbf{x}_t)^\omega \propto p_\theta(\mathbf{x}_t|c) \left[\frac{p_\theta(\mathbf{x}_t|c)}{p_\theta(\mathbf{x}_t)}\right] = \frac{p_\theta(\mathbf{x}_t|c)^{\omega+1}}{p_\theta(\mathbf{x}_t)^\omega}, \tag{3.11}$$

and, in the log space, we obtain

$$\log \tilde{p}_\theta(\mathbf{x}_t|c) = (\omega+1)\log p_\theta(\mathbf{x}_t|c) - \omega \log p_\theta(\mathbf{x}_t) + C, \tag{3.12}$$

which results in a sampling algorithm shown in Alg. 3.

3.2.2 Training and Sampling Improvements

Since the increase in popularity of DDPMs, a few ways have been proposed to speed up sampling, as simple DDPMs require more $T \geq 1000$ to learn the data distribution with the highest fidelity. Of these proposals, the most successful have been Denoising Diffusion Implicit Models (DDIMs), introduced in Song et al. [104], which we employ in our work. DDIMs generalize the standard diffusion process by allowing each step to

Algorithm 3 CFG Sampling [39]

Require: class label c, guidance strength ω

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for all t from T to 1 **do**
  $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\tilde{\boldsymbol{\varepsilon}} \leftarrow (\omega+1)\boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t, c) - \omega\boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t)$ # Two forward passes
  $\boldsymbol{\mu} \leftarrow \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\tilde{\boldsymbol{\varepsilon}}\right)$
  $\mathbf{x}_{t-1} \leftarrow \boldsymbol{\mu} + \sigma_t\boldsymbol{\varepsilon}$
end for
return $\mathbf{x}_0$

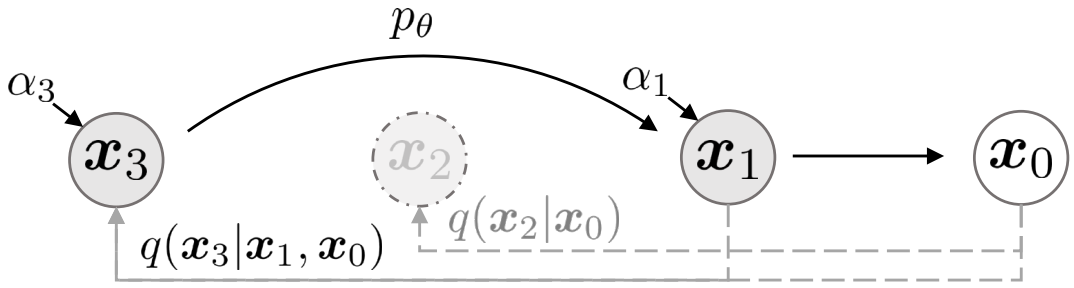

Figure 3.2: The DDIM graphical model for accelerated generation. Fig. 2 from Song et al. [104].

depend not only on the previous step but also on the original data point. This results in a non-Markovian diffusion process that, while maintaining the same training objective as DDPMs, significantly accelerates the sampling process. We show this graphical model in Fig. 3.2 (compare to DDPM in Fig. 3.1). The generative process in DDIMs is deterministic, allowing for high-quality samples in just a fraction of the steps required by traditional DDPMs. For a subset of steps T, where $s < t$, the step then becomes:

$$q_{\sigma,s<t}(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_s; \sqrt{\bar{\alpha}_s}\left(\frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\varepsilon_\theta(\mathbf{x}_t, t)}{\sqrt{\bar{\alpha}_t}}\right) + \sqrt{1-\bar{\alpha}_s-\sigma_t^2}\varepsilon_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I}) \tag{3.13}$$

where σ_t is a float controlling the stochasticity of the process and can be set so that the process becomes a DDPM. In particular, setting $\sigma_t = 0$, the process becomes fully deterministic, leading to faster and more consistent sample generation. For a given sample x_t, the next step in the reverse process can be obtained by:

$$\mathbf{x}_{t-1} = \underbrace{\sqrt{\alpha_{t-1}}\left(\frac{\mathbf{x}_t - \sqrt{1-\alpha_t}\varepsilon_\theta(\mathbf{x}_t, t)}{\sqrt{\alpha_t}}\right)}_{\text{predicted } x_0} + \underbrace{\sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot\varepsilon_\theta(\mathbf{x}_t, t)}_{\text{direction pointing to } x_t} + \underbrace{\sigma_t\varepsilon_t}_{\text{random noise}}, \tag{3.14}$$

where $\varepsilon \sim \mathcal{N}(0, I)$ is just Gaussian noise. Empirically, DDIMs have demonstrated the ability to generate samples up to 50 times faster than DDPMs, with only a minor trade-off in sample quality.

Another notable improvement is that of the β noise schedule. When initial DDPMs and DDIMs models converted the datapoint into pure noise *linearly*, Nichol and Dhariwal [85] found that this type of schedule converts images to noise too fast, therefore making the reverse process difficult to learn. Therefore, it was proposed to use a function that changes much slower towards the endpoints; this was achieved by using a *cosine* schedule, which is

$$\bar{\alpha}_t = \frac{f(t)}{f(0)}, \text{ and } f(t) = \cos\left(\frac{t/T + s}{1+s}\cdot\frac{\pi}{2}\right)^2, \tag{3.15}$$

and β_t is obtained from $\beta_t = 1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}$.

Finally, Hang et al. [34] notice that diffusion models are often slow to converge partly by virtue of conflicting optimisation directions between timesteps. To tackle this, they propose treating diffusion training as a multi-task learning problem and introduce a novel approach called the Min-SNR-γ loss weighting strategy. This implies the loss for a timestep t involving a sum over losses from multiple timesteps, that is

$$\mathcal{L} = \sum_{t=1}^{T} \omega_t \mathcal{L}_t(\theta), \tag{3.16}$$

where ω_t is the respective weight. Defining Signal to Noise Ratio at each of the diffusion steps as $\text{SNR}(t) = \frac{\alpha_t^2}{\sigma_t^2}$, the weight at each timestep is calculated as

$$\omega_t = \min\{\text{SNR}(t), \gamma\}, \tag{3.17}$$

with the weights being clamped to ensure that no timestep is given too much or too little emphasis. Here, γ is a predefined threshold that prevents the weight from becoming too large, which would overly prioritize certain timesteps at the expense of others.

3.2.3 Diffusion Model Disentanglement Schemes

Interpretable direction discovery for disentanglement in diffusion models refers to the process of identifying and isolating specific, meaningful latent directions within the model's representation space, which correspond to distinct and semantically understandable variations in the generated data, such as changes in object orientation, style, or attributes, thereby enabling controlled and interpretable manipulation of generated samples; due to the fact diffusion models are a relatively recent invention, rather few schemes have been proposed, which we briefly review here.

Kwon et al. [57] were the first to introduce the concept of an asymmetric reverse process, which discovers a *semantic latent space*—termed *h*-space — in pre-trained diffusion models. *h*-space is a new latent space derived from the bottleneck features of the CNN U-Net, which has the shape of an autoencoder, encapsulating high-level semantic information. The authors modify the reverse process by asymmetrically altering only certain components of the latent space. The method involves shifting the predicted noise ϵ_θ at each timestep in a controlled manner to adjust the attributes of the generated image. A small neural network is used to learn an implicit function that generates the required modifications in *h*-space for any given timestep and image feature. This function is trained to optimize the alignment of the generated image with the desired attribute. Zhang et al. [134] further this approach by proposing the first unsupervised and learning-based method to identify interpretable directions in *h*-space. By jointly optimizing these components, the model spontaneously discovers disentangled and interpretable directions. This approach is notable for its ability to maintain the fidelity of the generated samples by using a discriminator network, preventing the discovery of meaningless and destructive directions.

However, in our work, the most inspiration stems from the work of Haas et al. [32], who proposed a novel approach to discovering both global and local semantic directions within *h*-space for unconditional generation. Their method leverages Principal Component Analysis (PCA) to uncover global, interpretable directions, such as pose and gender, directly from the principal components of the bottleneck features in *h*-space. Through many passes of the network, the authors record the *h*-space activations and concatenate them, and then decompose them into n principal components for each of the timesteps. For boosting a certain attribute, at each t, the chosen component vector is added to the bottleneck activations, creating reproducible, albeit not necessarily fully disentangled, editing capabilities. This approach is especially attractive due to requiring

little modification for a state-of-the-art implementation of a diffusion model and does not require using an asymmetric process, which has not been proven to deliver better sample quality.

We acknowledge diffusion disentanglement schemes which are out of our reach due to the specifics of our model and compute constraints. Park et al. [87] introduce a method to derive local latent bases by leveraging the pullback metric associated with the encoding feature maps, allowing for precise image editing by moving within the latent space along these discovered basis vectors, but their method does not prove to provide greater results than that of Haas et al. [32] while being more expensive. Furthermore, two approaches have been introduced for text-prompted diffusion models: Li et al. [64] present a self-discovery approach that identifies interpretable latent directions in the h-space of diffusion models without requiring external classifiers or labelled datasets. Instead, their method uses a modified text prompt and the pre-trained model's internal representations to guide the discovery of concept vectors. On the other hand, Dalva and Yanardag [20] introduce NoiseCLR, an unsupervised contrastive learning-based framework for discovering interpretable directions in text-to-image diffusion models. NoiseCLR does not rely on text prompts or labelled data; instead, it uses a small set of unlabelled images from specific domains to discover latent directions that enable semantically meaningful and disentangled edits. The latter two may be applicable for 3D generative diffusion models which are being prompted by text.

Chapter 4

Methodology

In this chapter, we describe the methodology we employ to construct DART-IDE. First, we look at the arrangement used to construct an LSGM from SAOR using generative diffusion models and their architectures. Then, we discuss the proposed disentanglement scheme based on the PCA exploration of the h-space. Finally, we look at how the dataset used was constructed and the evaluation metrics that we have employed.

4.1 Generation Scheme

DART-IDE is based on two separate class-conditioned DDIM generators, the first responsible for the shape and articulation conditioning, and the second responsible for texture generation. Optionally, another generator, an addition to the first one, allows for manipulating the articulation of a set shape. These generators provide input for other modules of SAOR, where the input encoded by the scrapped modules would otherwise have been inferred from the input image.

4.1.1 Module Arrangement

We shall refer to the first generator as G_{features}, and the second as G_{texture}; optionally, we can use $G_{\text{articulation}}$ to manipulate the articulation separately. According to the description in Sect. 2.2, G_{features} generates $\phi_{\text{features}} \in \mathbb{R}^{512}$. However, instead of generating $\phi_{\text{texture}} \in \mathbb{R}^{512\times8\times8}$ with G_{texture} to feed into f_{t} MLPs, we bypass this by instead feeding an empty tensor $\phi_{\text{texture}} = \mathbf{0}^{512\times8\times8}$ to f_{t} and then rely on G_{texture} to generate $T \in \mathbb{R}^{3\times64\times64}$ images, which then are upscaled to the required $3\times256\times256$ resolution using an off-the-shelf pre-trained 4x super-resolution ESRGAN [118], and

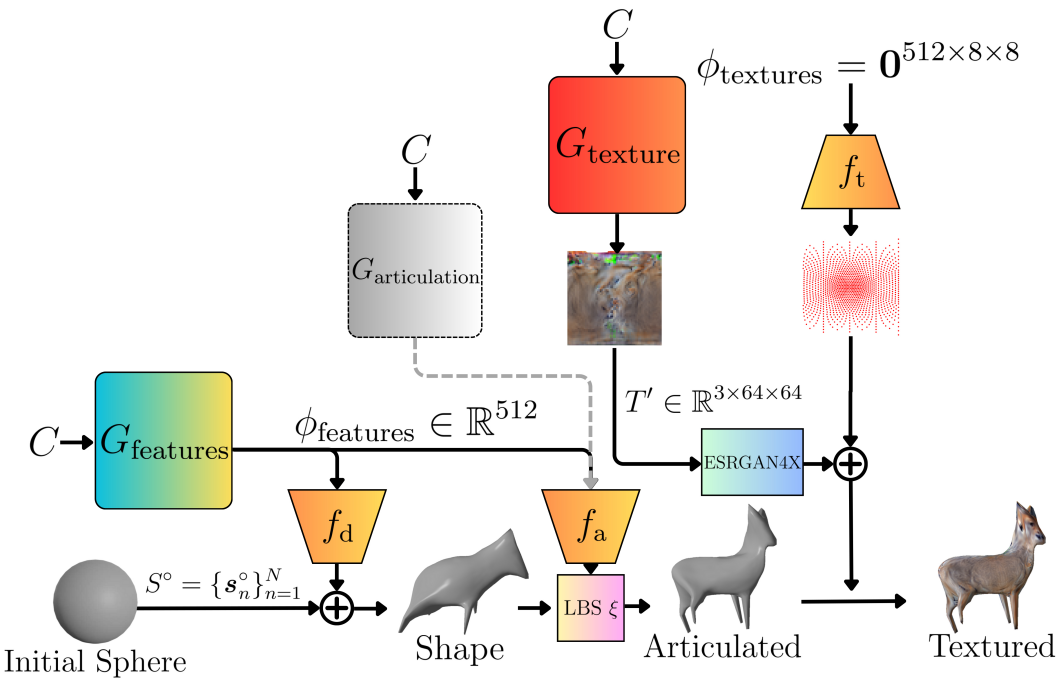

Figure 4.1: Overview of the DART-IDE model architecture (compare with architecture of SAOR in Fig. 2.1). It is composed of two class c conditioned DDIM models, where G_{features} determines the shape and articulation, and G_{texture} determines the texture; optionally, $G_{\text{articulation}}$ has the same architecture and weights as G_{features}, but may generate a different articulation. All parts of the generation are modular and independent, and all of the c labels used may be different. The generated ϕ_{features} vectors are fed into pre-trained SAOR modules, while the generated undersized texture T' is upscaled by a pre-trained ESRGAN, and then merged with a UV map from a filler tensor.

then doubled by flipping along the right edge to satisfy the symmetry. The mere propagation of ϕ_{features}, even composed of zeros, allows us to generate the required symmetry-satisfying UV map, required to associate specific vertices of the mesh with specific pixels of the texture, and we find that learning to generate T images directly yields results of much finer quality than generating ϕ_{texture}. The choice to work on 64 rather than 256 resolution images and then use a super-resolution model is motivated by computational constraints and is an instance of LSGMs itself. The complete architecture is shown in Fig. 4.1.1.

Therefore, G_{features} is trained on $(\phi_{\text{features}}, c)$ and G_{features} is trained on $(\texttt{downscale}(3 \times 64 \times 64, T), c)$ tuples respectively, obtained in the procedure described in Sect. 4.3.

4.1.2 Model Architectures

G_{features} generating 1D $\phi_{\text{features}} \in \mathbb{R}^{512}$ vectors means departing from traditional 2D convolutions in U-Net CNNs and working with 1 dimensional convolution operations, while G_{texture} employs a similar but more traditional 2D CNN architecture.
Both architectures use a bottleneck U-Net structure, where the input sample first gets its number of channels expanded and spatial resolution kept by the initial convolution in the layer `I`, then gradually gets reduced in spatial dimension and enlarged in the channel dimension by "down" blocks `D`, pass the bottleneck block `M`, and then undergo opposite transformations to the original spatial resolution in the "up" blocks `U`, to finally be reduced to the original channel number by the `F` convolution. In this work, we employ weight-standardised convolutions [91], which means that the weights in the convolutional filters are normalised by subtracting their mean and dividing by their standard deviation, that is,

$$\hat{W}_{i,j} = \frac{W_{i,j} - \mu_{W_{i,\cdot}}}{\sigma_{W_{i,\cdot}}}, \tag{4.1}$$

where $W_{i,j}$ are the original weights, $\mu_{W_{i,\cdot}}$ is the mean of the weights in the convolutional filter, and $\sigma_{W_{i,\cdot}}$ is the standard deviation of the weights in the convolutional filter. This approach works well in tandem with Group Normalisation [54] by reducing the variance of the input to subsequent layers. Each of the `D`, `M` and `U` blocks are composed of two convolutions, each followed by Group Normalisation [128] and SiLU activation [24]. Group normalisation normalises across groups of channels, improving training stability. The two convolutional layers are connected by a residual connection [35], which adds the input of the block to its output, mitigating the vanishing gradient problem. Furthermore, after each block, we employ the attention mechanism [116], which helps to focus on the relevant parts of the intermediate representation and is computed as

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{4.2}$$

where Q, K and V are the query, key and value matrices, respectively, derived from the input with three additional standard convolutions, and d_k is the dimensionality of the key. Furthermore, skip connections are introduced between `D` and `U` blocks of corresponding dimensions, ensuring that fine-grained details lost during downsampling are reintroduced in the upsampling phase. These skip connections *concatenate* (rather

than add) feature maps from the downsampling path with those in the upsampling path, providing the network with both high-level abstract features and low-level details.
To incorporate the class label c and the current timestep t into the network, we inject the condition *into each block* with a procedure similar to Feature-wise Linear Modulation [88]. The time information is embedded using a sinusoidal positional embedding (PE) [116], that is,

$$\mathrm{PE}(t) = \left[\sin\left(\frac{t}{10000^{2i/d}}, \cos\left(\frac{t}{10000^{2i/d}}\right)\right)\right], \tag{4.3}$$

where d is the dimension of the embedding, i indexes the dimensions, and $10000^{2i/d}$ controls the frequency, ensuring that each of the t's is represented uniquely while preserving relative temporal distance information through a periodic signal. The time embedding is passed through two linear MLPs with GELU [37] non-linearities. On the other hand, class information is learned using an embedding layer, followed by another MLP. These two embeddings are concatenated and passed through another MLP with a SiLU non-linearity, outputting a tensor with a dimension that is twice the desired block dimension. This tensor is chunked into two components - the first one is multiplied with the feature map to *scale* it, while the second chunk is added to the feature map to *shift* it, allowing the network to adaptively condition on temporal and class information. Combined, both models form state-of-the-art diffusion backbones, and in combination have 109 million parameters, comparable, to, for example, Base BERT [22] in NLP. Both used CNNs (simplified to residual blocks) and the activation dimensions in the forward propagation are shown in Fig. 4.1.2. We note that the models could have been made larger, but limit ourselves to having the largest models while keeping the training batch size to at least 64 and not running out of VRAM on an NVIDIA V100 GPU, and have in mind that generally in U-Nets, increasing the depth (number of levels) enhances the network's ability to capture multi-scale features and improves gradient flow due to skip connections, which is usually more beneficial than increasing the number of channels [77].

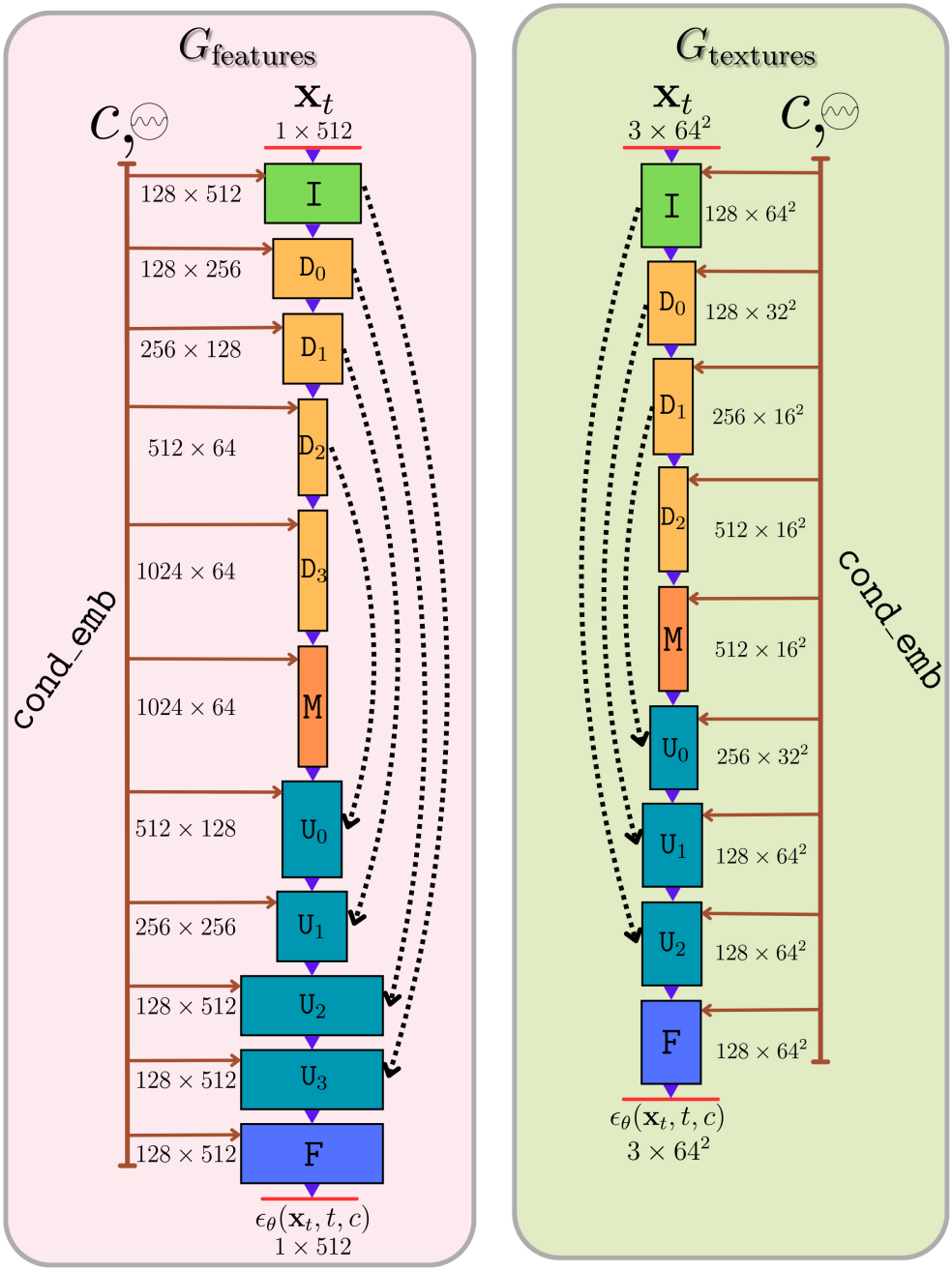

Figure 4.2: Vertical views of G_{features} and G_{texture} backbone U-Nets, with the former using 1D and the latter 2D convolutions, annotated with the activation dimensions, where channel number is followed by the spatial dimension. Each model is composed of an input convolution, downsampling blocks, bottleneck block, upsampling blocks and the final convolution. The dotted arrows show the concatenating skip connections, while the brown arrows show the injection of class c label through trainable embeddings and time t through sinusoidal embeddings into every block.

4.2 Disentanglement Scheme

Haas et al. [32] base their diffusion disentanglement approach on recording and analysing h-space (bottleneck) activations and then performing PCA on them. However, this approach only works with diffusion models that are not trained to generate samples according to some given class. We extend this approach to class-conditional diffusion models, and this technique works for any kind of backbone neural network.

Due to the fact that in CFG diffusion there are $2T$ passes to generate every sample (recall Alg. 3), we propose to generate n final samples and then *only* record the h-space activations for the unconditional pass $\varepsilon_\theta(\mathbf{x}_t, t)$, saving a separate tensor for each timestep t, and then finding the direction of largest variance in each timestep; with this, we hope to discover directions *invariant* to the class that can be amplified or reduced. Specifically, having trained the DDIM using $T_{\text{features}} = 100$ for G_{features}, we record bottleneck activations using $n = 1,024$, to obtain a tensor of shape $[n, T, \texttt{bottleneck channels dim}, \texttt{bottleneck spatial dim(s)}]$. Considering $V = 10$ principal components, we use the Incremental PCA algorithm [70] to obtain principal components scaled by singular values of shape

$$[V, T, \texttt{bottleneck channels dim}, \texttt{bottleneck spatial dim(s)}],$$

by the virtue of

$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T, \tag{4.4}$$

where $\mathbf{X}$ is the matrix of h-space activations, $\mathbf{U}$ contains the principal components, Σ is the diagonal matrix of singular values, and $\mathbf{V}$ contains the right singular vectors. The principal components are ordered by the amount of variance they explain in the data. Having picked a component corresponding to some semantic concept $v \in V$ and making the DDIM process deterministic (as outlined in Subsect. 3.2.2) to make the generated samples constant, we can add the vector of the chosen principal component scaled by a constant scalar Δh at the required time step $t \in T$ to provide meaningful semantic changes to the generated meshes.

4.3 Dataset Preparation

For training our models, we designed a custom dataset, composed of the subset of quadruped animal images from iNaturalist [44] that SAOR had been trained, as described in Sect. 2.2. Having generated the meshes from the complete set of considered

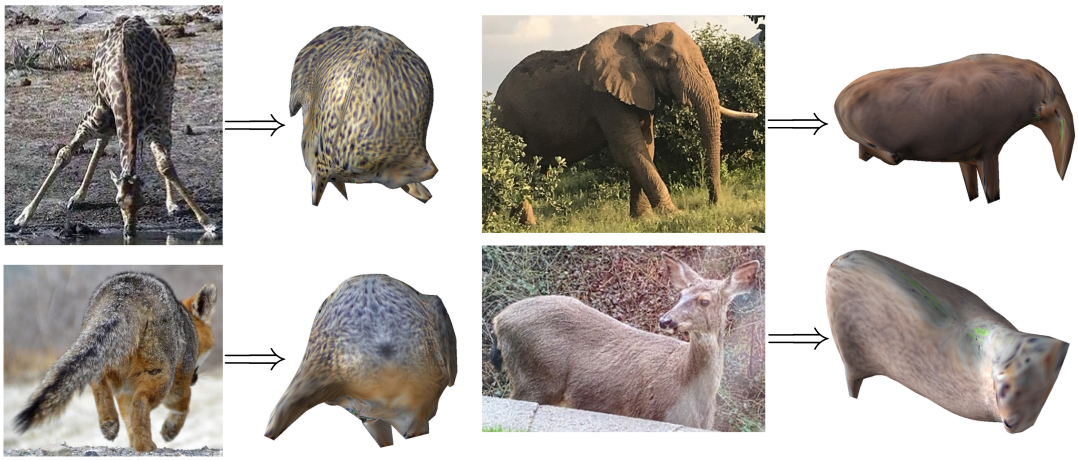

Figure 4.3: Cases of SAOR failing to predict the shape of the animal.

quadruped images, we devise a set of criteria on which the output meshes from SAOR fail. Rejection criteria include obscured limbs, unextended or strongly bent limbs, the animal having lay down, or the animal being pictured in such a way that the full length of its body is not sufficiently shown, such as when the animal is photographed from the front or back. We show a few cases in which SAOR fails to predict the shape in Fig. 4.3 as examples of images that had been rejected.

We also reject black and white images for modelling the texture distribution more truthfully. With these criteria, from the initial 31k images, we pick 14,352 training samples by hand, rejecting around 56% of the original images, and using 16 animal classes; the least represented category has 437 images, while the most represented one has 1,529 images. For these images, we save the global latent ϕ_{features}, texture image $T \in \mathbb{R}^{3\times256\times256}$ and the mesh object, which within itself contains the UV map. We note that for optimal dataset creation, the image edges were cut down to 102% of the provided bounding box, contrary to the procedure in SAOR training; furthermore, to check whether there had been any mistakes in the provided bounding boxes, a quick KNN clustering check between the cut down and original images with SIFT [78] extracted features was performed with a 100% accuracy. Afterwards, as the SAOR encoders require a square input image, the edges had been filled with a white border of the required size, and the image was scaled down to the required $3 \times 64 \times 64$ resolution using the Lanczos algorithm [80]. Additionally, we note that the pose network f_{p} of SAOR (recall the architecture in Fig. 2.1) was disabled during the dataset generation, resulting in meshes that are by default encoded in an approximately constant coordinate system, always centred approximately around the centre of mass of the animal and by

default rendered from a horizontal side view and only being slightly distributed along the elevation angle. This was done for the shape generation evaluation metric, described in Sect. 4.4, to perform more reliably.

4.4 Evaluation Metrics and Training Details

Being the first to generate articulated and textured meshes in a generative context, our methodology needs a unique approach, for which we develop a new metric. Except for class distributions, we assume that the generated object shape and texture are independent. To avoid a situation where the training set is just memorised, we use *early stopping* [30], that is, while training, we periodically check our metrics and save the model only when the metric has been improved. On the other hand, to avoid the double descent phenomenon characteristic of models this large [99], which is characterised by a nonmonotonic behaviour in which the test error initially decreases, then increases and finally decreases again as the model capacity grows, we make sure that the *patience* parameter is sufficiently large, meaning that a significant number of training iterations are allowed without improvement in the metric before stopping the training, ensuring the model has enough time to explore improvements before halting.

For shape generation training, our training procedure consists of learning the distribution of ϕ_{features} and then periodically generating a certain number of textureless meshes, as in Fig. 4.1.1, but *without* G_{texture}, and then using our metric for early stopping. We propose a novel metric, which we coin the AMPCD. First, consider simple *Chamfer distance*, which, for two sets of points S_1 and S_2, bilaterally finds the pairs of points that are the closest in the opposite set. These distances are then summed and the Chamfer distance d_{CD} is defined as the sum of these distances across both sets:

$$d_{\text{CD}}(S_1,S_2) = \frac{1}{|S_1|}\sum_{x\in S_1}\min_{y\in S_2}\|x-y\|_2^2 + \frac{1}{|S_2|}\sum_{y\in S_2}\min_{x\in S_1}\|x-y\|_2^2. \tag{4.5}$$

Then, for two *sets of meshes* $A = \{K_i\}_{i=1}^{i=|A|}$ and $B = \{J_i\}_{i=1}^{i=|J|}$, we define AMPCD to correspond to how well, on average, each of the shapes approximated at least one of the shapes in the ground truth set and vice versa, picking the shape that got approximated best as a reference:

$$d_{\text{AMPCD}}(A,B) = \frac{1}{|A|}\sum_{i=1}^{|A|}\min_{j\in\{1,\dots,|B|\}} d_{\text{CD}}(K_i,J_j) + \frac{1}{|B|}\sum_{j=1}^{|B|}\min_{i\in\{1,\dots,|A|\}} d_{\text{CD}}(K_i,J_j). \tag{4.6}$$

The first term is crucial for understanding if the generated outputs are close to any of the ground truth meshes, while the second one evaluates how well each ground truth mesh is represented in the generated set, ensuring that the generated set is diverse enough to cover the ground truth set. We note that a similar metric could have been devised using the Hausdorff distance, which is more sensitive to outlier points of the mesh than the Chamfer distance, but did not find this useful for our dataset. Having turned off f_{p} during the ground truth mesh generation for the dataset, we find that AMPCD captures the quality of generated meshes highly efficiently - the smaller the metric, the better the quality.

For texture generation, we use Fréchet Inception Distance (FID) [38], which has become the de facto standard metric for evaluating generated image quality and has been the metric of choice in all papers mentioned in Chapter 3. FID measures the similarity between the distribution of real images and the distribution of generated images by comparing their statistics in the feature space of a pre-trained Inception CNN [109]. Specifically, it computes the Fréchet distance between two multivariate Gaussians fitted to the feature embeddings of real and generated images. These embeddings are derived from the coding layer of the Inception network, capturing the high-level features of the images. The FID score is calculated as follows:

$$d_{\text{FID}}((m,C),(m_w,C_w)) = \|m-m_w\|_2^2 + \text{Tr}(C+C_w-2(CC_w)^{1/2}), \tag{4.7}$$

where m and C are the mean and covariance of the generated images' embeddings, and m_w and C_w are the mean and covariance of the real images' embeddings. Lower FID values indicate that the generated images are more similar to the real images, with a value of zero representing perfect similarity.

As mentioned in Sect. 4.3, the distribution of classes in the dataset is imbalanced. Therefore, we take special care when computing the metrics. When calculating AMPCD for G_{features} at each evaluation interval, having generated 64 meshes for each of the 16 classes, we take a *weighted* average throughout the classes to get the final result, to reflect the distribution of classes in the dataset; i.e. the metric for each class is weighted by $\frac{\text{\# examples in class}}{\text{\# total examples}}$, and they are added together. On the other hand, as FID calculation is even more expensive and could not be parallelised more, the number of generated examples for each class is $\frac{\text{\# examples in class}\times 1024}{\text{\# total examples}}$ while training G_{texture}, and we take the FID with the whole training set.

We summarize the training and procedural hyperparameters for both models in Tab. 4.1.

The decision to reduce DDIM sampling timesteps and U-Net Dimension Multipliers for G_{texture} is largely determined by the computational restraints and the need to still get a reliable estimate of the metrics while sampling much larger T's takes a lot longer than ϕ_{features}. Controlled by the computational restraints too is the number of generated samples for checking the metrics; albeit ideally, we would like to generate the number of samples equal to the number of examples in the dataset, our arrangement is still statistically significant to a satisfactory extent compared to acceptable approaches in other work, e.g. for calculating their FIDs, Nichol and Dhariwal [85] generate only 10,000 images to compare to 1,281,167 images in ImageNet64x64 [114]. We vary our learning rate in two linear segments, from the initial one to the warmup one to the final one, determined to work empirically. The γ parameter is set from the original work [34]. Note, however, that in Chapter 5, in order to improve the quality and find the best combination between sample fidelity and diversity, we do sweep through CFG parameters of conditional scaling (ω) and probability of dropping class information during training (p_{uncond}), as outlined in Subsect. 3.2.1.

Parameter	G_{features}	G_{texture}
β **schedule**	cosine	
Timesteps	1000	
Sampling Timesteps	100	50
Initial Channels	128	
U-Net Dimension Multipliers	(1, 2, 4, 8)	(1, 2, 4)
γ	5	
Batch Size	64	
Initial LR	0.0003	
Warmup Finish LR	0.0002	
Final LR	0.0001	
LR # Warmup Epochs	15	90
# Samples to Check Metric	1024	
Metric Patience	15	12
Warmup Epochs	15	90

Table 4.1: Training parameters for G_{features} and G_{texture}. Warmup Epochs refers to the number of epochs we train for prior to starting to check the required method for early stopping, while other parameters have been explained in previous sections.

Chapter 5

Results and Discussion

In this chapter, we evaluate the architectural decisions made throughout the development of DART-IDE, including a hyperparameter sweep to improve model performance. We also present the results obtained, accompanied by an assessment of the AMPCD metric and our contributions to the disentanglement of the latent space. Each section is paired with relevant discussions and analyses to provide deeper insights into our findings. In our experiments, we utilised FP16 mixed precision training [82] to accelerate training and used the Adam optimiser [50]. The total computational effort for these experiments amounted to approximately 700 GPU hours on an NVIDIA V100 GPU.

5.1 Architectural and Experimental Considerations

In this section, we discuss the architectural choices made for class-conditioning of DART-IDE and the hyperparameter optimisation.

5.1.1 Classifier-Based or Classifier-Free Guidance?

Having implemented the models according to the general methodology in Chapter 4 [1], we begin by making an architectural decision between separate classifier-based guidance and CFG, described in Subsect. 3.2.1. We are quick to discover that due to the small size of our dataset, even strong ResNet classifiers with Attention, Batch Normalisation [45], Dropout [107], and horizontal flip augmentations for T are unable to classify noisy ϕ_{feature} vectors and T images to a satisfactory extent, achieving around 50%

[1] Using the Hugging Face DDPM tutorial (www.huggingface.co/blog/annotated-diffusion) and Phil Wang's DDPM repo (www.github.com/lucidrains/denoising-diffusion-pytorch) as starting points.

and 53% Top-1 accuracies, respectively, with an 80-20 train-validation split. This is much less than what Dhariwal and Nichol [23] had achieved with their used ImageNet classifier (64.9% Top-1 accuracy); due to this, and also wanting to be able to make quick adjustments to the dataset without the need of retraining the classifier, we proceed with CFG.

5.1.2 Conditional Scaling and Condition Dropping Probability Sweep

Having chosen CFG as the means of conditional generation for DART-IDE, we sweep through values of guidance strength ω (looking for a fine balance between increasing the fidelity of the generated samples at the expense of diversity and balancing fidelity and diversity) and the probability of condition dropping p_{uncond} (finding the most effective balance between the model's capacity for generating conditional versus unconditional samples) while using an extensive grid sweep with 3 random seeds. The chosen values are based on observations from the original work of Ho and Salimans [39], and we are checking between moderate values, taking into account that both AMPCD and FID encourage great quality and punish too little diversity in the generated samples.

ω \ p_{uncond}	**0.1**	**0.3**	**0.5**
0.5	$2.38 \pm 0.22 \times 10^{-3}$	$2.07 \pm 0.31 \times 10^{-3}$	$2.27 \pm 0.06 \times 10^{-3}$
3.0	$1.70 \pm 0.07 \times 10^{-3}$	$1.75 \pm 0.03 \times 10^{-3}$	$1.86 \pm 0.05 \times 10^{-3}$

Table 5.1: The results of G_{features} hyperparameter sweep with early stopping on AMPCD using 3 random seeds, with the best result in red.

ω \ p_{uncond}	**0.1**	**0.3**	**0.5**
0.5	30.55 ± 3.40	32.77 ± 0.91	35.98 ± 4.27
3.0	29.48 ± 3.09	31.72 ± 0.72	35.57 ± 7.24

Table 5.2: The results of G_{texture} hyperparameter sweep with early stopping on FID and using 3 random seeds, with the best result in red.

We observe that for both ϕ_{features} and T image generation, more aggressive guidance $\omega = 3.0$ generally delivers better early stopping results, while the best combination is

obtained with $p_{\text{uncond}} = 0.1$. This implies that the data strongly has a conditional structure, and benefits from the model staying closer to the desired conditional distribution; furthermore, conditional information is highly informative and necessary for accurate generation, while the unconditional component of the training should be minimized.

5.2 Shape and Articulation Conditioning and Generation

In this section, we first look at the untextured meshes generated from DART-IDE, and briefly assess our new metric, AMPCD. Textured meshes are discussed in Sect. 5.4.

5.2.1 Generated Meshes

The hyperparameter-optimised G_{features} module of DART-IDE provides a strong model to generate highly diverse fine-quality articulated shapes, examples of which are shown in Fig. 5.1. The generation of an untextured mesh takes hundreds of milliseconds on a single NVIDIA RTX3060 GPU and tens of milliseconds on a V100 GPU.

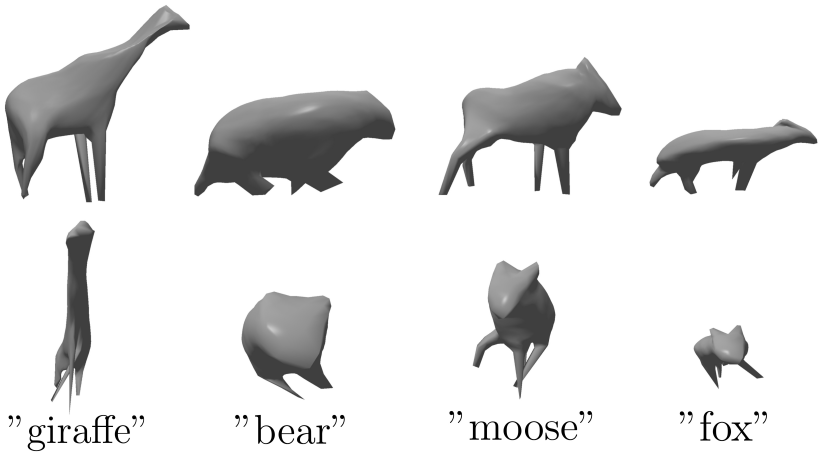

Figure 5.1: Untextured meshes from 4 classes generated with DART-IDE. Compare with original meshes from SAOR in Appendix

5.2.2 Assessment of AMPCD as a Metric

A generative ML metric must demonstrate robustness in being directly relevant to the specific task or application for which the generative model is designed, sensitive

enough to detect subtle differences in quality between different outputs, and, especially for generative models, be able to reflect differences in the diversity of the generated outputs, all of which have been demonstrated by FID [47]. Furthermore, Carlini et al. [16] have shown that even though there is no "overfitting" as such in diffusion training methods, DDPMs may *memorise* the training data given a dataset too small and too many training steps, which makes the use of early stopping with a diversity accounting metric a great scheme. As such, we seek to demonstrate a desired behaviour similar to that obtained with our novel metric, AMPCD. We show that the second term in the definition of AMPCD delivers a similar effect to FID by comparing the loss and AMPCD development throughout epochs in Fig. 5.2, and demonstrate the progression of the generation trade-off between quality and diversity throughout epochs in Fig. 5.3. We also show the class-wise distribution of AMPCD with the best hyperparameters during the early stopping epoch, compared with the number of examples in each class, in Fig. 5.4. Therefore, we see that AMPCD provides a strong metric, which accounts for both generation fidelity and diversity and does not require separate consideration of both aspects, contrary to the minimum matching distance and coverage metrics in Lei et al. [61].

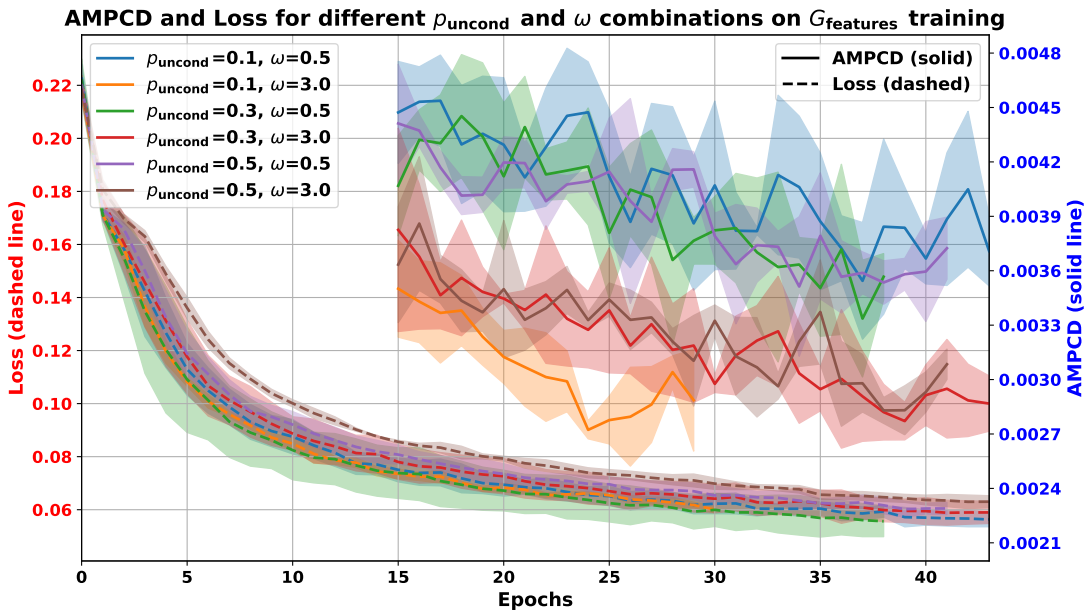

Figure 5.2: AMPCD and the training loss using training settings in Tab. 4.1 throughout the sweep in Tab. 5.1 for G_{features}, averaged throughout 3 random seeds. As the DDPM loss still decreases, AMPCD starts increasing, reflecting the decrease in diversity.

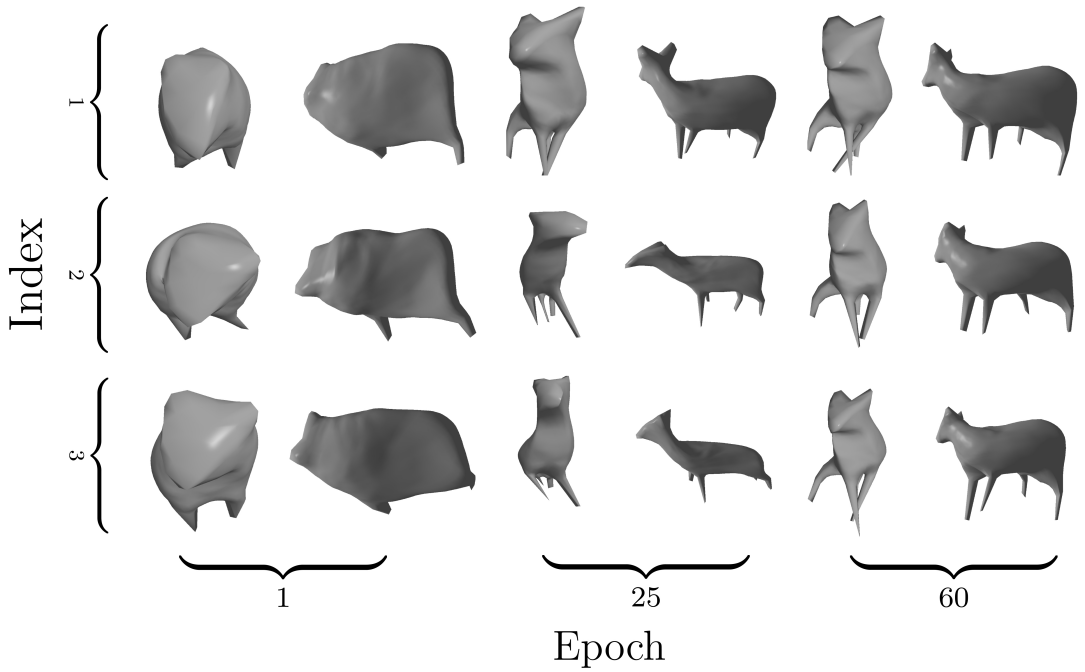

Figure 5.3: Shape and articulation generation progression for class $C = 1$ (”roe deer”) throughout the epochs with $\omega = 3.0$ and $p_{\text{uncond}} = 0.1$, but typical of all combinations. Early epochs deliver poor quality and little diversity, near early stopping epochs deliver great quality and great diversity, while very late epochs deliver great quality with very little diversity, very often “memorising” the most typical poses in the dataset.

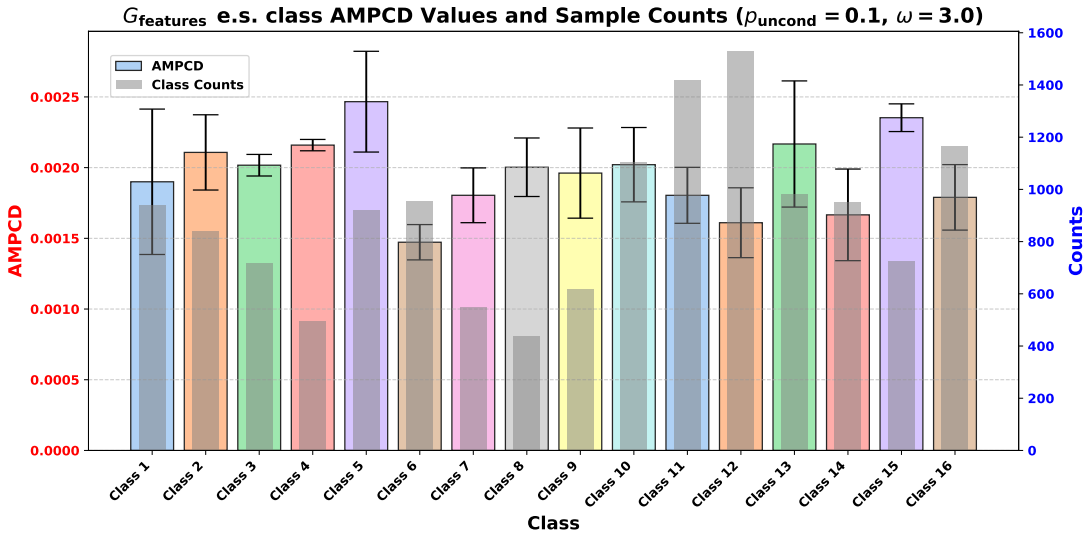

Figure 5.4: AMPCD distribution on the early stopping epoch with 3 seeds on $p_{\text{uncond}} = 0.1$ and $\omega = 3.0$. More class examples may not correspond to a better AMPCD.

5.3 Texture Conditioning and Generation

G_{texture} is a typical class-conditional image DDIM, and, given the limits on model, compute, and dataset size, achieved FID 29.48 with 1024 images, which drops to ∼12 when increasing the amount of generated images to 10k, is very respectable. Generating a single T takes around a minute on an RTX3060 GPU and around 10 seconds on an A100 GPU, and benefits greatly from parallelisation/batching. The examples T' are shown in Fig. 5.5 (green artefacts are typical of textures generated by SAOR), and we show examples of non-upscaled and upscaled T's in App. A. We note that we could have performed a similar analysis for FID as for AMPCD in Subsect. 5.2.2, but do not, as the former is a well-established metric.

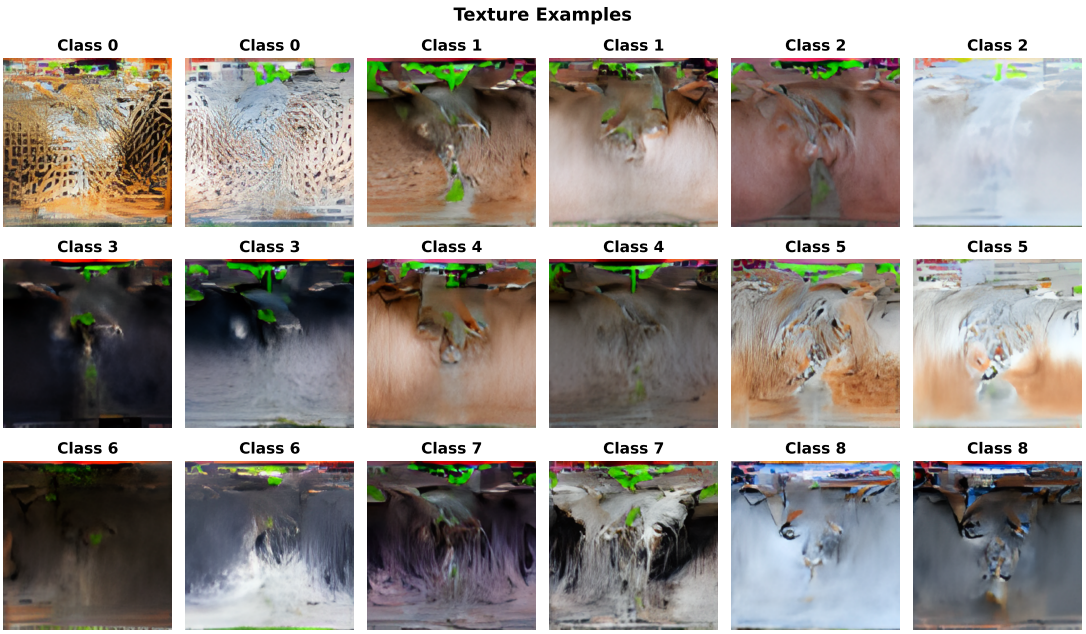

Figure 5.5: Textures T generated with G_{texture} of DART-IDE and upscaled with ESRGAN.

5.4 Full Results

When combined, G_{features}, G_{texture} and, optionally, $G_{\text{articulation}}$ make DART-IDE into a powerful generator, where all three can occur independently, and the class conditioning of different latents can be different. We show examples of generated meshes in Fig. 5.6, and possible applications of independent generation of texture and articulation in Fig. 5.7 and Fig. 5.8. The complete generation times depend on DDIM times (Sects. 5.2 and 5.3), while inference times for the SAOR modules being used are negligible.

Figure 5.6: Example textured meshes generated by DART-IDE. Green artefacts, typically seen on the bottom "seam" of the mesh, are typical of textures generated by SAOR.

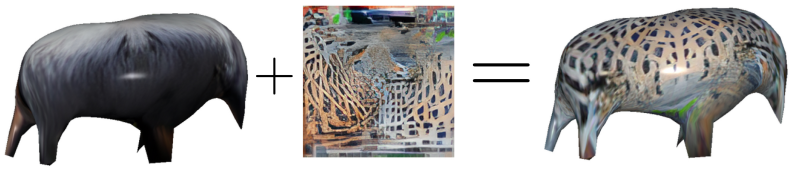

Figure 5.7: Independent texture generation and transfer from one animal class (giraffe) to the other (elephant).

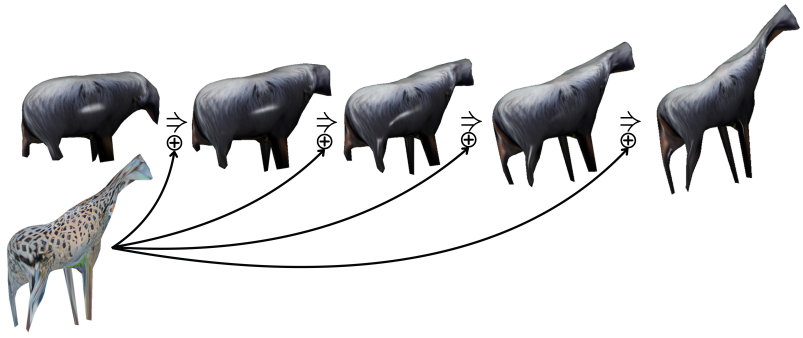

Figure 5.8: Independent articulation generation and transfer from one animal class (giraffe) to the other (elephant), gradually mixing in inputs from $G_{\text{articulation}}(c = \text{giraffe})$.

5.5 Interpretable Direction Discovery

We perform interpretable direction discovery on the optimised version of DART-IDE with the methodology described in Subsect. 4.2. We discover that although unconditional passes make up just a fraction of all forward propagations, the principal discovered components of the h-space influence the generated meshes rather consistently across classes, and, having frozen the forward pass in addition to the modulation, we see that the components include elongation and the angle of the head ("uprightness" and "sitedness"), volume of the animal, the side of the body that the curvature is in, the width of the body, limb proportion, and their joint angles. However, as expected from the work of Haas et al. [32], the interpretable directions discovered are not fully disentangled from the other components, and even more so than in the unconditional model. We show the editing that we can perform with the components discovered in Fig. 5.9.

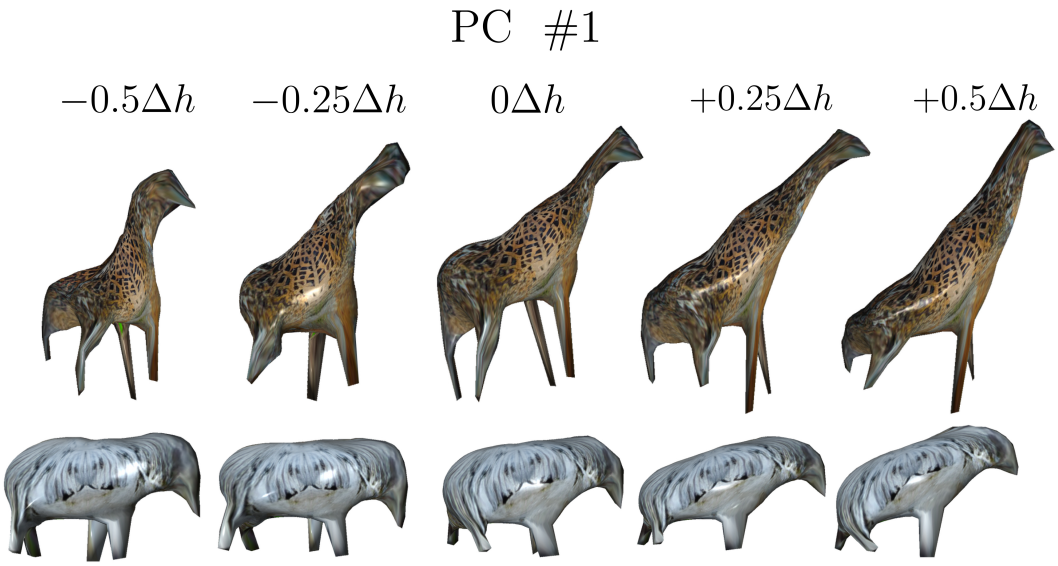

Figure 5.9: Editing performed on the meshes with PCA of the h-space in the unconditional passes, where Principal Component 1 corresponds to "uprightness"/"sitedness"; that is, with more Δh subtracted, the animal raising its back feet and pulling in its neck, while the opposite direction makes it sit down and elongate its neck. This change happens consistently throughout different classes. The coefficients of ± 0.25 and ± 0.5 were determined to work reasonably well empirically. We too, note that the editing is not detached from other attributes, e.g. the arrangement of the legs, and show that adding the same amount of Δh may qualitatively influence the meshes to different extents.

Chapter 6

Conclusions

In this final chapter, we first summarise the key findings of the project and the key contributions we have made. Next, we acknowledge the limitations of our undertaking and then draw some guidelines for possible future research avenues.

6.1 Summary

In this thesis, we reviewed the recent progress in 3D computer vision and diffusion models and then developed and introduced a novel framework, DART-IDE, for generating articulated and textured 3D meshes using DDIM LSGMs, with control over shape, articulation, and texture. The framework extends the capabilities of existing models, particularly the SAOR model, to independently generate and control the shape, articulation, and texture of 3D objects. We have developed state-of-the-art U-Net backbones for our models, and through rigorous experimentation, we determined the optimal architecture and hyperparameters, achieving high fidelity and diversity in the generated outputs. We have demonstrated that DART-IDE can generate high-quality, diverse, and customisable 3D content, being the first model of its kind in the controllability of different modalities. The versatility of DART-IDE was further demonstrated by successful applications of independent texture transfer and articulation adjustment across various classes of animals. Furthermore, we extend the state-of-the-art in interpretable direction discovery in the latent space of class-conditional diffusion models, extending an existing h-space method to identify meaningful, albeit not fully disentangled, semantic directions that can be used to predictably modify generated meshes. We too are the first to demonstrate h-space-based disentanglement on a modality other than images (here, meshes).
We introduced a new metric, AMPCD, specifically designed to assess the quality and

diversity of generated 3D articulated meshes and address the limitations of existing evaluation methods. This metric has proven to be effective in guiding early stopping during training, balancing between the quality and diversity of the generated shapes. Last but not least, we introduce an image dataset, which would help train a model similar to DART-IDE, but using another backbone rather than SAOR.
In summary, as we expect diffusion models to keep rising in importance (being used as LSGMs too) in 3D generation, we hope our work makes a valuable contribution in being able to generate largely controllable meshes of great quality, a new metric, and advances in interpretable direction discovery.

6.2 Limitations & Future Work

While DART-IDE has made strides, there are limitations, a few of which we note here to be addressed in future work. First, we note that one can very rarely find large diffusion models being trained on datasets containing less than 20k samples, which we did here, limiting the fidelity to which the distribution can be represented (since we were only using the useful *subsets* of the data SAOR had been trained on). Furthermore, the dataset had been rather biased, often portraying the animal with a curvature of the body to one side rather than the other. As such, the next iteration would have to obtain additional training data from other sources. For example, it would be really interesting to retrain a larger version of SAOR with an application to broader categories, e.g. including non-animal objects, and then repeat our procedure. Furthermore, as discussed in Subsect. 2.2, when the code is released for 3DFauna [69] and Farm3D [46], it would be interesting to try them as backbones instead of SAOR, as they should produce better meshes from more varying angles. As far as disentanglement goes, we could further expand our analysis to *class-specific* principal components, but do not because of the lack of time; here, we would expect class-specific components to be less entangled with other attributes when editing. However, the direction that we would find most exciting with the current setup, but with more compute, would be to use CLIP [93] to transform DART-IDE into a text-promptable model by conditioning its generative processes on text embeddings. This would allow us to generate and manipulate 3D models directly from natural language descriptions, making 3D content creation more intuitive and accessible. Furthermore, this would allow us to use advanced text-prompted diffusion disentanglement techniques (as described at the end of Subsect. 3.2.3), and we would obtain the first editable text-to-mesh, rather than text-to-NeRF, generator.

Bibliography

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning Representations and Generative Models for 3D Point Clouds, 2018. URL https://arxiv.org/abs/1707.02392.

[2] Tomer Amit, Tal Shaharbany, Eliya Nachmani, and Lior Wolf. SegDiff: Image Segmentation with Diffusion Probabilistic Models, 2022. URL https://arxiv.org/abs/2112.00390.

[3] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2(3):283–310, Jan 1989. ISSN 1573-1405. doi: 10.1007/BF00158167. URL https://doi.org/10.1007/BF00158167.

[4] Mehmet Aygün and Oisin Mac Aodha. SAOR: Single-View Articulated Object Reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10382–10391, June 2024.

[5] Song Bai and Jie Li. Progress and Prospects in 3D Generative AI: A Technical Overview including 3D human, 2024. URL https://arxiv.org/abs/2401.02620.

[6] Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Gregoire Mialon, Yuandong Tian, Avi Schwarzschild, Andrew Gordon Wilson, Jonas Geiping, Quentin Garrido, Pierre Fernandez, Amir Bar, Hamed Pirsiavash, Yann LeCun, and Micah Goldblum. A cookbook of self-supervised learning, 2023. URL https://arxiv.org/abs/2304.12210.

[7] Kuldeep R Barad, Andrej Orsula, Antoine Richard, Jan Dentler, Miguel Olivares-Mendez, and Carol Martinez. GraspLDM: Generative 6-DoF Grasp Synthesis

using Latent Diffusion Models, 2023. URL https://arxiv.org/abs/2312.11243.

[8] Reza Bayat. A Study on Sample Diversity in Generative Models: GANs vs. Diffusion Models, 2023. URL https://openreview.net/forum?id=BQpCuJoMykZ.

[9] Reiner Birkl, Diana Wofk, and Matthias Müller. MiDaS v3.1 – A Model Zoo for Robust Monocular Relative Depth Estimation, 2023. URL https://arxiv.org/abs/2307.14460.

[10] Sam Bond-Taylor, Adam Leach, Yang Long, and Chris G. Willcocks. Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7327–7347, November 2022. ISSN 1939-3539. doi: 10.1109/tpami.2021.3116668. URL http://dx.doi.org/10.1109/TPAMI.2021.3116668.

[11] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23 (11):1222–1239, 2001. doi: 10.1109/34.969114.

[12] Matej Božić and Marko Horvat. A Survey of Deep Learning Audio Generation Methods, 2024. URL https://arxiv.org/abs/2406.00146.

[13] Andrew Brock, Theodore Lim, J. M. Ritchie, and Nick Weston. Generative and Discriminative Voxel Modeling with Convolutional Neural Networks, 2016. URL https://arxiv.org/abs/1608.04236.

[14] P. Burt and E. Adelson. The Laplacian Pyramid as a Compact Image Code. *IEEE Transactions on Communications*, 31(4):532–540, 1983. doi: 10.1109/TCOM.1983.1095851.

[15] Antoine Caillon and Philippe Esling. RAVE: A variational autoencoder for fast and high-quality neural audio synthesis, 2021. URL https://arxiv.org/abs/2111.05011.

[16] Nicholas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramèr, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting

Training Data from Diffusion Models, 2023. URL https://arxiv.org/abs/2301.13188.

[17] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient Geometry-aware 3D Generative Adversarial Networks, 2022. URL https://arxiv.org/abs/2112.07945.

[18] Stanley H. Chan. Tutorial on Diffusion Models for Imaging and Vision, 2024. URL https://arxiv.org/abs/2403.18103.

[19] Dan Cireşan, Ueli Meier, Jonathan Masci, and Jürgen Schmidhuber. Multi-column deep neural network for traffic sign classification. *Neural Networks*, 32:333–338, 2012. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2012.02.023. URL https://www.sciencedirect.com/science/article/pii/S0893608012000524. Selected Papers from IJCNN 2011.

[20] Yusuf Dalva and Pinar Yanardag. NoiseCLR: A Contrastive Learning Approach for Unsupervised Discovery of Interpretable Directions in Diffusion Models, 2023. URL https://arxiv.org/abs/2312.05390.

[21] Larry S. Davis. A survey of edge detection techniques. *Computer Graphics and Image Processing*, 4(3):248–270, 1975. ISSN 0146-664X. doi: https://doi.org/10.1016/0146-664X(75)90012-X. URL https://www.sciencedirect.com/science/article/pii/0146664X7590012X.

[22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019. URL https://arxiv.org/abs/1810.04805.

[23] Prafulla Dhariwal and Alex Nichol. Diffusion Models Beat GANs on Image Synthesis, 2021. URL https://arxiv.org/abs/2105.05233.

[24] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-Weighted Linear Units for Neural Network Function Approximation in Reinforcement Learning, 2017. URL https://arxiv.org/abs/1702.03118.

[25] Noureen Fatima, Ali Shariq Imran, Zenun Kastrati, Sher Muhammad Daudpota, and Abdullah Soomro. A Systematic Literature Review on Text Generation

Using Deep Neural Network Models. *IEEE Access*, 10:53490–53503, 2022. doi: 10.1109/ACCESS.2022.3174108.

[26] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, Jan 2005. ISSN 1573-1405. doi: 10.1023/B:VISI.0000042934.15159.49. URL https://doi.org/10.1023/B:VISI.0000042934.15159.49.

[27] R. Fergus, P. Perona, and A. Zisserman. Weakly supervised scale-invariant learning of models for visual recognition. *International Journal of Computer Vision*, 71(3):273–303, 03 2007. URL https://www.proquest.com/scholarly-journals/weakly-supervised-scale-invariant-learning-models/docview/1113643283/se-2. Copyright - Springer Science + Business Media, LLC 2006; Last updated - 2023-11-26.

[28] Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao Zhang. Sdm-net: Deep generative network for structured deformable mesh, 2019. URL https://arxiv.org/abs/1908.04520.

[29] Shubham Goel, Angjoo Kanazawa, and Jitendra Malik. Shape and Viewpoint without Keypoints, 2020. URL https://arxiv.org/abs/2007.10982.

[30] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL http://www.deeplearningbook.org. Book in preparation for MIT Press.

[31] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks, 2014. URL https://arxiv.org/abs/1406.2661.

[32] René Haas, Inbar Huberman-Spiegelglas, Rotem Mulayoff, Stella Graßhof, Sami S. Brandt, and Tomer Michaeli. Discovering Interpretable Directions in the Semantic Latent Space of Diffusion Models, 2024. URL https://arxiv.org/abs/2303.11073.

[33] Song Han. Lecture Slides in MIT 6.5940, Massachusetts Institute of Technology, 2023.

[34] Tiankai Hang, Shuyang Gu, Chen Li, Jianmin Bao, Dong Chen, Han Hu, Xin Geng, and Baining Guo. Efficient Diffusion Training via Min-SNR Weighting Strategy, 2024. URL https://arxiv.org/abs/2303.09556.

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition, 2015. URL https://arxiv.org/abs/1512.03385.

[36] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, 2015. URL https://arxiv.org/abs/1502.01852.

[37] Dan Hendrycks and Kevin Gimpel. Gaussian Error Linear Units (GELUs), 2023. URL https://arxiv.org/abs/1606.08415.

[38] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium, 2018. URL https://arxiv.org/abs/1706.08500.

[39] Jonathan Ho and Tim Salimans. Classifier-Free Diffusion Guidance, 2022. URL https://arxiv.org/abs/2207.12598.

[40] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models, 2020. URL https://arxiv.org/abs/2006.11239.

[41] Emiel Hoogeboom, Victor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant Diffusion for Molecule Generation in 3D, 2022. URL https://arxiv.org/abs/2203.17003.

[42] Tao Hu, Liwei Wang, Xiaogang Xu, Shu Liu, and Jiaya Jia. Self-supervised 3d mesh reconstruction from single images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6002–6011, June 2021.

[43] G. M. Hunter and K. Steiglitz. Operations on Images Using Quad Trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(02), apr 1979. ISSN 1939-3539. doi: 10.1109/TPAMI.1979.4766900.

[44] iNaturalist. inaturalist. https://www.inaturalist.org. Accessed: 2024-11-12.

[45] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, 2015. URL https://arxiv.org/abs/1502.03167.

[46] Tomas Jakab, Ruining Li, Shangzhe Wu, Christian Rupprecht, and Andrea Vedaldi. Farm3D: Learning Articulated 3D Animals by Distilling 2D Diffusion, 2024. URL https://arxiv.org/abs/2304.10535.

[47] Sadeep Jayasumana, Srikumar Ramalingam, Andreas Veit, Daniel Glasner, Ayan Chakrabarti, and Sanjiv Kumar. Rethinking FID: Towards a Better Evaluation Metric for Image Generation, 2024. URL https://arxiv.org/abs/2401.09603.

[48] Tero Karras, Samuli Laine, and Timo Aila. A Style-Based Generator Architecture for Generative Adversarial Networks, 2019. URL https://arxiv.org/abs/1812.04948.

[49] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering, 2023. URL https://arxiv.org/abs/2308.04079.

[50] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, 2017. URL https://arxiv.org/abs/1412.6980.

[51] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014. URL https://arxiv.org/abs/1312.6114.

[52] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders, 2019.

[53] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment Anything, 2023. URL https://arxiv.org/abs/2304.02643.

[54] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big Transfer (BiT): General Visual Representation Learning, 2020. URL https://arxiv.org/abs/1912.11370.

[55] Taku Komura. Lecture Slides in Computer Animation and Visualisation (INFR11067), University of Edinburgh, February 2020.

[56] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[57] Mingi Kwon, Jaeseok Jeong, and Youngjung Uh. Diffusion Models already have a Semantic Latent Space, 2023. URL https://arxiv.org/abs/2210.10960.

[58] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-Based Learning Applied to Document Recognition. In *Proceedings of the IEEE*, number 11, pages 2278–2324, 1992.

[59] Yann Lecun, Sumit Chopra, Raia Hadsell, Marc Aurelio Ranzato, and Fu Jie Huang. *"A tutorial on energy-based learning"*. MIT Press, 2006.

[60] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521 (7553):436–444, May 2015. ISSN 1476-4687. doi: 10.1038/nature14539. URL https://doi.org/10.1038/nature14539.

[61] Jiahui Lei, Congyue Deng, Bokui Shen, Leonidas Guibas, and Kostas Daniilidis. NAP: Neural 3D Articulation Prior, 2023. URL https://arxiv.org/abs/2305.16315.

[62] Chenghao Li, Chaoning Zhang, Atish Waghwase, Lik-Hang Lee, Francois Rameau, Yang Yang, Sung-Ho Bae, and Choong Seon Hong. Generative AI meets 3D: A Survey on Text-to-3D in AIGC Era, 2024. URL https://arxiv.org/abs/2305.06131.

[63] Chun-Liang Li, Manzil Zaheer, Yang Zhang, Barnabas Poczos, and Ruslan Salakhutdinov. Point Cloud GAN, 2018. URL https://arxiv.org/abs/1810.05795.

[64] Hang Li, Chengzhi Shen, Philip Torr, Volker Tresp, and Jindong Gu. Self-Discovering Interpretable Diffusion Latent Directions for Responsible Text-to-Image Generation, 2024. URL https://arxiv.org/abs/2311.17216.

[65] Jing Li, Di Kang, Wenjie Pei, Xuefei Zhe, Ying Zhang, Zhenyu He, and Linchao Bao. Audio2Gestures: Generating Diverse Gestures from Speech Audio with Conditional Variational Autoencoders, 2021. URL https://arxiv.org/abs/2108.06720.

[66] Jun Li, Chenyang Zhang, Wei Zhu, and Yawei Ren. A Comprehensive Survey of Image Generation Models Based on Deep Learning. *Annals of Data Science*, Jun 2024. ISSN 2198-5812. doi: 10.1007/s40745-024-00544-1. URL https://doi.org/10.1007/s40745-024-00544-1.

[67] Xiaoyu Li, Qi Zhang, Di Kang, Weihao Cheng, Yiming Gao, Jingbo Zhang, Zhihao Liang, Jing Liao, Yan-Pei Cao, and Ying Shan. Advances in 3D Generation: A Survey, 2024. URL https://arxiv.org/abs/2401.17807.

[68] Zehui Li, Yuhao Ni, William A V Beardall, Guoxuan Xia, Akashaditya Das, Guy-Bart Stan, and Yiren Zhao. DiscDiff: Latent Diffusion Model for DNA Sequence Generation, 2024. URL https://arxiv.org/abs/2402.06079.

[69] Zizhang Li, Dor Litvak, Ruining Li, Yunzhi Zhang, Tomas Jakab, Christian Rupprecht, Shangzhe Wu, Andrea Vedaldi, and Jiajun Wu. Learning the 3D Fauna of the Web, 2024. URL https://arxiv.org/abs/2401.02400.

[70] Jongwoo Lim, David Ross, Ruei-sung Lin, and Ming-Hsuan Yang. Incremental learning for visual tracking. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2004. URL https://proceedings.neurips.cc/paper_files/paper/2004/file/f21e255f89e0f258accbe4e984eef486-Paper.pdf.

[71] Jian Liu, Xiaoshui Huang, Tianyu Huang, Lu Chen, Yuenan Hou, Shixiang Tang, Ziwei Liu, Wanli Ouyang, Wangmeng Zuo, Junjun Jiang, and Xianming Liu. A Comprehensive Survey on 3D Content Generation, 2024. URL https://arxiv.org/abs/2402.01166.

[72] Minghua Liu, Ruoxi Shi, Linghao Chen, Zhuoyang Zhang, Chao Xu, Xinyue Wei, Hansheng Chen, Chong Zeng, Jiayuan Gu, and Hao Su. One-2-3-45++: Fast Single Image to 3D Objects with Consistent Multi-View Generation and 3D Diffusion, 2023. URL https://arxiv.org/abs/2311.07885.

[73] Pengkun Liu, Yikai Wang, Fuchun Sun, Jiafang Li, Hang Xiao, Hongxiang Xue, and Xinzhou Wang. Isotropic3D: Image-to-3D Generation Based on a Single CLIP Embedding, 2024. URL https://arxiv.org/abs/2403.10395.

[74] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot One Image to 3D Object, 2023. URL https://arxiv.org/abs/2303.11328.

[75] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning, 2019. URL https://arxiv.org/abs/1904.01786.

[76] Yixin Liu, Kai Zhang, Yuan Li, Zhiling Yan, Chujie Gao, Ruoxi Chen, Zhengqing Yuan, Yue Huang, Hanchi Sun, Jianfeng Gao, Lifang He, and Lichao Sun. Sora: A Review on Background, Technology, Limitations, and Opportunities of Large Vision Models, 2024. URL https://arxiv.org/abs/2402.17177.

[77] Vincent Loos, Rohit Pardasani, and Navchetan Awasthi. Demystifying the Effect of Receptive Field Size in U-Net Models for Medical Image Segmentation, 2024. URL https://arxiv.org/abs/2406.16701.

[78] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov 2004. ISSN 1573-1405. doi: 10.1023/B:VISI.0000029664.99615.94. URL https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[79] Calvin Luo. Understanding Diffusion Models: A Unified Perspective, 2022. URL https://arxiv.org/abs/2208.11970.

[80] B. N. Madhukar and R. Narendra. "Lanczos Resampling for the Digital Processing of Remotely Sensed Images". In Veena S. Chakravarthi, Yasha Jyothi M. Shirur, and Rekha Prasad, editors, *Proceedings of International Conference on VLSI, Communication, Advanced Devices, Signals and Systems and Networking (VCASAN-2013)*, pages 403–411, India, 2013. Springer India. ISBN 978-81-322-1524-0.

[81] David Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. Henry Holt and Co., Inc., New York, NY, USA, 1982. ISBN 0716715678.

[82] Paulius Micikevičius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed Precision Training, 2018. URL https://arxiv.org/abs/1710.03740.

[83] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, 2020. URL https://arxiv.org/abs/2003.08934.

[84] Tom Monnier, Matthew Fisher, Alexei A. Efros, and Mathieu Aubry. Share With Thy Neighbors: Single-View Reconstruction by Cross-Instance Consistency, 2022. URL https://arxiv.org/abs/2204.10310.

[85] Alex Nichol and Prafulla Dhariwal. Improved Denoising Diffusion Probabilistic Models, 2021. URL https://arxiv.org/abs/2102.09672.

[86] Michael Niemeyer and Andreas Geiger. GIRAFFE: Representing Scenes as Compositional Generative Neural Feature Fields, 2021. URL https://arxiv.org/abs/2011.12100.

[87] Yong-Hyun Park, Mingi Kwon, Jaewoong Choi, Junghyo Jo, and Youngjung Uh. Understanding the Latent Space of Diffusion Models through the Lens of Riemannian Geometry, 2023. URL https://arxiv.org/abs/2307.12868.

[88] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual Reasoning with a General Conditioning Layer, 2017. URL https://arxiv.org/abs/1709.07871.

[89] Walter H. L. Pinaya, Petru-Daniel Tudosiu, Jessica Dafflon, Pedro F da Costa, Virginia Fernandez, Parashkev Nachev, Sebastien Ourselin, and M. Jorge Cardoso. Brain Imaging Generation with Latent Diffusion Models, 2022. URL https://arxiv.org/abs/2209.07162.

[90] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D Diffusion, 2022. URL https://arxiv.org/abs/2209.14988.

[91] Siyuan Qiao, Huiyu Wang, Chenxi Liu, Wei Shen, and Alan Yuille. Micro-Batch Training with Batch-Channel Normalization and Weight Standardization, 2020. URL https://arxiv.org/abs/1903.10520.

[92] Lynn H. Quam. Hierarchical warp stereo. In Martin A. Fischler and Oscar Firschein, editors, *Readings in Computer Vision*, pages 80–86. Morgan Kaufmann, San Francisco (CA), 1987. ISBN 978-0-08-051581-6. doi: https://doi.org/10.1016/B978-0-08-051581-6.50015-5. URL https://www.sciencedirect.com/science/article/pii/B9780080515816500155.

[93] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision, 2021. URL https://arxiv.org/abs/2103.00020.

[94] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-Shot Text-to-Image Generation, 2021. URL https://arxiv.org/abs/2102.12092.

[95] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating Diverse High-Fidelity Images with VQ-VAE-2, 2019. URL https://arxiv.org/abs/1906.00446.

[96] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows, 2016. URL https://arxiv.org/abs/1505.05770.

[97] Lawrence G. Roberts. *Machine perception of three-dimensional solids*. PhD thesis, Massachusetts Institute of Technology, USA, 1963.

[98] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. URL https://arxiv.org/abs/1505.04597.

[99] Rylan Schaeffer, Mikail Khona, Zachary Robertson, Akhilan Boopathy, Kateryna Pistunova, Jason W. Rocks, Ila Rani Fiete, and Oluwasanmi Koyejo. Double Descent Demystified: Identifying, Interpreting and Ablating the Sources of a Deep Learning Puzzle, 2023. URL https://arxiv.org/abs/2303.14151.

[100] Sandeep Singh Sengar, Affan Bin Hasan, Sanjay Kumar, and Fiona Carroll. Generative Artificial Intelligence: A Systematic Review and Applications, 2024. URL https://arxiv.org/abs/2405.11029.

[101] Farooq Sijal Shaqwi, Lukman Audah, Mustafa Hamid Hassan, Mohammed Ahmed Jubair, Mohd Helmy Abd Wahab, and Salama A. Mostafa. A concise review of deep learning deployment in 3d computer vision systems. In *2021 4th International Symposium on Agents, Multi-Agent Systems and Robotics (ISAMSR)*, pages 157–160, 2021. doi: 10.1109/ISAMSR53229.2021.9567757.

[102] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition, 2015. URL https://arxiv.org/abs/1409.1556.

[103] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics, 2015. URL https://arxiv.org/abs/1503.03585.

[104] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising Diffusion Implicit Models, 2022. URL https://arxiv.org/abs/2010.02502.

[105] Yang Song and Stefano Ermon. Generative Modeling by Estimating Gradients of the Data Distribution, 2020. URL https://arxiv.org/abs/1907.05600.

[106] Pratul P. Srinivasan, Stephan J. Garbin, Dor Verbin, Jonathan T. Barron, and Ben Mildenhall. Nuvo: Neural UV Mapping for Unruly 3D Representations, 2023. URL https://arxiv.org/abs/2312.05283.

[107] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

[108] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep High-Resolution Representation Learning for Human Pose Estimation, 2019. URL https://arxiv.org/abs/1902.09212.

[109] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions, 2014. URL https://arxiv.org/abs/1409.4842.

[110] Richard Szeliski. Computer vision algorithms and applications, 2011. URL http://dx.doi.org/10.1007/978-1-84882-935-0.

[111] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. DreamGaussian: Generative Gaussian Splatting for Efficient 3D Content Creation, 2024. URL https://arxiv.org/abs/2309.16653.

[112] Shubham Tulsiani, Nilesh Kulkarni, and Abhinav Gupta. Implicit Mesh Reconstruction from Unannotated Image Collections, 2020. URL https://arxiv.org/abs/2007.08504.

[113] Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based Generative Modeling in Latent Space, 2021. URL https://arxiv.org/abs/2106.05931.

[114] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel Recurrent Neural Networks, 2016. URL https://arxiv.org/abs/1601.06759.

[115] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural Discrete Representation Learning, 2018. URL https://arxiv.org/abs/1711.00937.

[116] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, 2023. URL https://arxiv.org/abs/1706.03762.

[117] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. Caltech-ucsd birds 200. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[118] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images, 2018. URL https://arxiv.org/abs/1804.01654.

[119] Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. SGPN: Similarity Group Proposal Network for 3D Point Cloud Instance Segmentation, 2019. URL https://arxiv.org/abs/1711.08588.

[120] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Chen Change Loy, Yu Qiao, and Xiaoou Tang. ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks, 2018. URL https://arxiv.org/abs/1809.00219.

[121] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. ProlificDreamer: High-Fidelity and Diverse Text-to-3D Generation

with Variational Score Distillation, 2023. URL https://arxiv.org/abs/2305.16213.

[122] Cheng Wen, Baosheng Yu, Rao Fu, and Dacheng Tao. Patch-Wise Point Cloud Generation: A Divide-and-Conquer Approach, 2023. URL https://arxiv.org/abs/2307.12049.

[123] Lilian Weng. What are Diffusion Models?, 2021. URL https://lilianweng.github.io/posts/2021-07-11-diffusion-models/.

[124] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T. Freeman, and Joshua B. Tenenbaum. Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling, 2017. URL https://arxiv.org/abs/1610.07584.

[125] Shangzhe Wu, Christian Rupprecht, and Andrea Vedaldi. Unsupervised Learning of Probably Symmetric Deformable 3D Objects from Images in the Wild, 2020. URL https://arxiv.org/abs/1911.11130.

[126] Shangzhe Wu, Tomas Jakab, Christian Rupprecht, and Andrea Vedaldi. DOVE: Learning Deformable 3D Objects by Watching Videos, 2022. URL https://arxiv.org/abs/2107.10844.

[127] Shangzhe Wu, Ruining Li, Tomas Jakab, Christian Rupprecht, and Andrea Vedaldi. Magicpony: Learning articulated 3d animals in the wild, 2023. URL https://arxiv.org/abs/2211.12497.

[128] Yuxin Wu and Kaiming He. Group Normalization, 2018. URL https://arxiv.org/abs/1803.08494.

[129] Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. Perspective Transformer Nets: Learning Single-View 3D Object Reconstruction without 3D Supervision, 2017. URL https://arxiv.org/abs/1612.00814.

[130] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion Models: A Comprehensive Survey of Methods and Applications, 2024. URL https://arxiv.org/abs/2209.00796.

[131] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop, 2016. URL https://arxiv.org/abs/1506.03365.

[132] Maciej Zamorski, Maciej Zieba, Piotr Klukowski, Rafal Nowak, Karol Kurach, Wojciech Stokowiec, and Tomasz Trzcinski. Adversarial autoencoders for compact representations of 3d point clouds, 2019. URL https://arxiv.org/abs/1811.07605.

[133] Jiahui Zhang, Fangneng Zhan, Muyu Xu, Shijian Lu, and Eric Xing. Fregs: 3d gaussian splatting with progressive frequency regularization, 2024. URL https://arxiv.org/abs/2403.06908.

[134] Zijian Zhang, Luping Liu, Zhijie Lin, Yichen Zhu, and Zhou Zhao. Unsupervised Discovery of Interpretable Directions in h-space of Pre-trained Diffusion Models, 2023. URL https://arxiv.org/abs/2310.09912.

[135] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection, 2017. URL https://arxiv.org/abs/1711.06396.

Appendix A

Comparison of Original and Upscaled Textures

We compare non-upscaled texture images from the G_{textures} part of DART-IDE with those upscaled using ESRGAN 4X [120] in Fig. A.1.

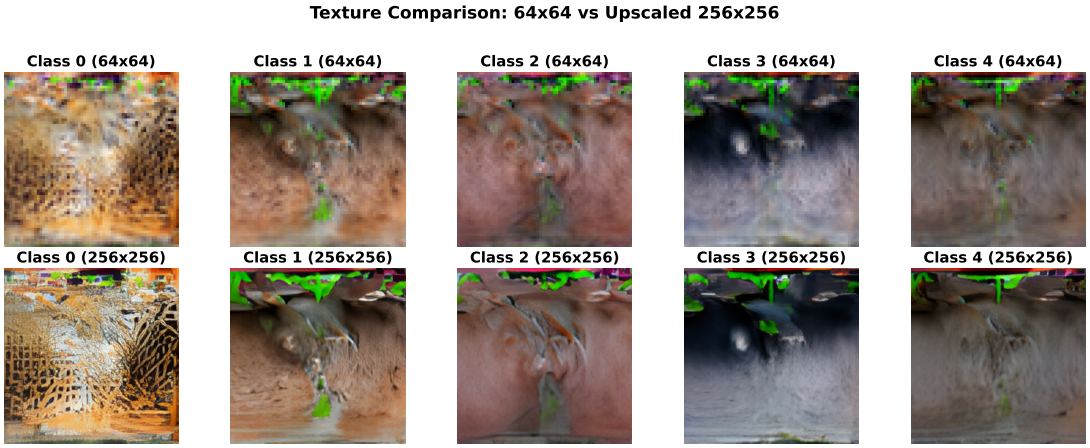

Figure A.1: Top - original $3 \times 64 \times 64$ textures. Bottom - upscaled $3 \times 256 \times 256$ textures.