ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN

COMPUTER SCIENCE AND ENGINEERING

CICLO 35

Settore Concorsuale: 09/H1 - Sistemi di elaborazione delle informazioni
Settore Scientifico Disciplinare: ING-INF/05 - Sistemi di elaborazione delle informazioni

---

Understanding 3D structures from Sketches

Presentata da:
Gianluca BERARDI

Coordinatore di Dottorato:
Ilaria BARTOLINI

Supervisore:
Prof. Luigi DI STEFANO

ESAME FINALE ANNO 2023

UNIVERSITÀ DI BOLOGNA

Abstract

Facoltà di Ingegneria ed Architettura
Dipartimento di Informatica - Scienza e Ingegneria

Dottorato di Ricerca

Understanding 3D structures from Sketches

by Gianluca Berardi

Sketches are a unique way to communicate: drawing a simple sketch does not require any training, sketches convey information that is hard to describe with words, they are powerful enough to represent almost any concept, and nowadays, it is possible to draw directly from mobile devices. Motivated from the unique characteristics of sketches and fascinated by the human ability to imagine 3D objects from drawings, this thesis focuses on automatically associating geometric information to sketches. The main research directions of the thesis can be summarized as obtaining geometric information from freehand scene sketches to improve 2D sketch-based tasks and investigating Vision-Language models to overcome 3D sketch-based tasks limitations.

The first part of the thesis concerns geometric information prediction from scene sketches improving scene sketch to image generation and unlocking new creativity effects. The thesis proceeds showing a study conducted on the Vision-Language models embedding space considering sketches, line renderings and RGB renderings of 3D shape to overcome the use of supervised datasets for 3D sketch-based tasks, that are limited and hard to acquire. Following the obtained observations and results, Vision-Language models are applied to Sketch Based Shape Retrieval without the need of training on supervised datasets. We then analyze the use of Vision-Language models for sketch based 3D reconstruction in an unsupervised manner. In the final chapter we report the results obtained in an additional project carried during the PhD, which has lead to the development of a framework to learn an embedding space of neural networks that can be navigated to get ready-to-use models with desired characteristics.

Contents

Chapter 1

Introduction

"A rough or unfinished drawing or painting, often made to assist in making a more finished picture". Its definition is actually very simple but a sketch has many forms and uses, it can represent an object from the real or a fantasy world, convey a concept or a sentiment, it can be elevated to art.

Sketching can be seen as an old, simple and **universal way to communicate** between human beings: drawing a sketch is simple for a human, it does not require any training and his understanding rely on common real world knowledge. Compared to learning a natural language like "English", there is no need to study to understand a sketch. Text is the main way people use to **search for content online, like images and videos**, a highly impactful computer vision task. However, natural language struggles to communicate certain information, like **the position or details of an object**, while drawing, instead, represent a very easy and natural way to convey such information. When other common image domains are considered, like realistic images or photos, sketches stand out for their **abstract, imprecise and sparse nature**, while still being a powerful enough representation to **communicate almost any idea or concept**. Nowadays, drawings is also possible from everyday mobile devices like smartphones and tablets, making **digital sketches easily accessible and drawing always an available option** .

The main characteristics that make sketches a valuable research direction can be summarized as follows: 1) a sketch can substitute or flank text in the computer vision task of searching for content online, like images and videos, conveying information that is difficult to describe with text, 2) mobile devices make digital sketching always an available option, obtaining ready-to-use data, 3) people don't have to learn to draw, it is easy and fast, 4) a sketch is an abstract, simple but very powerful way to communicate.

After motivating investigation in computer vision about sketches, the research directions pursued in the thesis are introduced in the following and progressing with the thesis it will be possible to note how the discussed motivational points will relate to most of the sketch tasks that will be presented.

A simple sketch is a good way to represent a single object or a scene. Looking at such a sketch, as humans, we are able to understand it, even if it is abstract and not precise. We can not only understand it, but we can depict some sort of image of the object or scene in our mind that is rich in information. We can think about colours, we can make objects move or perform actions and

we can think about their 3D geometry. This is possible because we have previous experience with many objects and scenes of the real world. This experience makes humans able to create in their mind 3D representations of objects or scenes depicted in a 2D image, like a sketch. The investigation of the replication of this behaviour, i.e. being able to automatically associate geometric information to a sketch, is the main objective of this thesis.

Considering the introduced concepts, the recent success of machine learning approaches in this research field is not too hard to justify. Obtaining a 3D representation of an object from a 2D image is an ill posed problem, information is missing. Humans take the missing information from previous experience and machine learning imitates this behaviour with a training phase on a large amount of data. For this reason, in this thesis, deep learning solutions are applied.

The thesis work focuses on the presented objective investigating two research questions.
1) Is it possible to obtain geometric information from a freehand scene sketch and use it to improve 2D sketch-based tasks? Many 2D sketch-based tasks exist, for example, nowadays we can retrieve images [1] or videos [2] and generate realistic colourful images by simply drawing a sketch [3][4]. While generating realistic images from sketches has a strong creative and artistic application, sketch-based retrieval has strong advantages compared to text retrieval, it is much easier to convey the position or the details of objects.

The geometry of a scene conveys information about how the objects are related to each other and with the environment of the scene. This helps to better understand the scene itself, therefore helping to improve many different kind of tasks. In chapter 3, deep learning algorithms are applied to extract geometric information from scene sketches and show how this information helps to improve the generation of RGB images and give access to new creative effects.

2) Can Vision-Language models be applied to sketches to overcome 3D sketch-based tasks limits? In the literature, two of the most investigated 3D sketch-based tasks are sketch-based shape retrieval [5][6] and 3D shape reconstruction from sketches [7] [8]. The sketch-based retrieval advantages in the 2D domain extend to the 3D domain, since, compared to other retrieval modalities, sketching is a simpler and more precise way to find what users are exactly looking for. Drawing as a way to generate 3D shapes is a powerful tool. Nowadays we still rely on complicated modelling software to obtain a 3D model of a new object and their use is limited to professionals. Obtaining desired 3D shapes drawing a simple sketch makes modelling much more accessible, opening to non-professional users the opportunity to create their own 3D shapes.

Recent solutions to these tasks [5][9] rely on deep learning techniques and make use of {sketch, 3D shape} paired datasets for training. Datasets with freehand sketches are hard to acquire, to obtain a {sketch, 3D shape} pair, a person is asked to draw a sketch for one or multiple renderings of a 3D shape. Available datasets [5] contain a limited amount of examples and just a few classes, strongly limiting the potential of supervised methods. Many works [10] [11] [12] [7] rely on synthetic sketches (or line renderings) as an alternative to freehand sketches, using {synth. sketch, 3D shape} paired datasets. These sketches can be easily obtained from the 3D shapes of objects, so it is possible to obtain a {synth sketch, 3D shape} dataset starting just from a dataset of 3D shapes. However they are precise and do not have the characteristics of freehand sketches, which present

a high variation: they could be highly abstract, their lines are often not precise, the stroke width may change and they are drawn with different styles by different people. For this reason, models trained on synthetic sketches suffer from a domain gap when applied to freehand sketches.

In this thesis an alternative approach to the data problem is proposed: the use of vision-language models is investigated to overcome the domain gap between freehand sketches and other image domains like synthetic sketches or RGB renderings. Vision-language models have shown impressive abilities to generalize to new domains on vision tasks [13]. They are trained on very large {text, image} datasets with millions of examples that show many different object and scene classes. If these models are capable to generalize to sketches, it could be possible to use these models as feature extractors directly, instead of learning them on {sketch, 3D shape} pairs, eliminating the need for this kind of paired datasets. Also, the limited amount of classes on available {sketch, 3D shape} paired datasets is not a problem when using vision-language models, they are already aware of many different classes. In this thesis, a study on a vision-language model, CLIP[13], is presented in Chapter 4, where an investigation on the CLIP embedding space is conducted comparing objects and views between different domains. Following the results and observations obtained from the study, the application of vision-language models on sketch-based 3D retrieval and reconstruction is investigated in Chapter 4 Section 4.2 and 4.3 respectively.

In parallel to the work on the main topic, together with a PhD student colleague, an investigation on another research direction has been pursued. This project is presented in the last chapter of the thesis, Chapter 5. In this project, intrigued by works about the redundancy of trained neural network weights, like the Lottery Ticket Hypothesis [14], the possibility to treat the weights of trained neural networks as they were redundant as the data these same networks are trained to process is investigated. In particular, in this project, the embedding space of neural network weights is learned in an encoder-decoder fashion, showing how, after the training phase, it is possible to navigate such a space through interpolation or optimization to obtain ready-to-use neural network instances with desired characteristics.

Chapter 2

A brief walk through the Sketch literature

A sketch is a simple way to communicate used by humans to express almost any kind of concept. Sketches have been studied for a long time in the computer vision and computer graphics literature and many works before the deep learning advent have been published [15] [16] [17] [18] [19] [20]. In the last decade, the combination of machine learning and mobile devices increased the interest and diffusion of sketches in the scientific community. This section is mainly focused on recent works, that were the most impactful on the topics presented in the thesis.

2.1 Types of Sketches

Different types of sketches exist. It is important to distinguish between synthetic sketches and freehand sketches. Synthetic sketches are artificially and automatically obtained from images [21] or 3D shapes [22], the latter also called line rendering. Freehand sketches may be professional [23], with very accurate and precise lines that well represent a real object, or amateur [24], simple sketches that anyone can draw without the need of any drawing skill, abstract with imprecise lines and different possible styles. This thesis will focus on amateur freehand sketches to progress the research in favor of all kind of users, independently from the drawing capacity. In the following, the term freehand sketches will always refer to the amateur type.

2.2 Sketch Representations

A person draws a sketch starting with a single, first stroke and then adds more strokes that all together form the final drawing. This is a specific characteristic of a sketch which is conducive of diverse forms of representation.

A sketch can be visualized as a raster and sparse image, where for many pixels the sketch does not define a value, these pixels correspond to the background and have a default value. This representation is the most common in the literature, it treats a sketch as a sparse image and makes it possible to apply to sketches the same architectures studied for images in general.

The Convolutional Neural Nerwork (CNN) [25][26] is an architecture specifically defined for images where, considering a layer, every filter works locally, striding over the image and producing a feature map as output. This architecture is very successful in extracting high quality features from images and nowadays its use is widespread. Lately, also Vision Transformers (ViTs) [27], a Transformer architecture [28] adapted to manage images, showed the ability to extract features from images on pair or even surpassing CNNs. Most of the papers that will be treated in this section are based on this image representation and corresponding architectures, they will be discussed afterwards divided per task.

A different way to represent a sketch is with a sequence of strokes, where the points of every stroke are defined as a sequence of coordinates. A common format to store sketches as sequences is the SVG, and when it came to input a sequence of strokes to neural networks Stroke3 [29] and Stroke5 [30] formats were defined.

Recurrent Neural Network (RNN) [31][32] is a well-known architecture that manages this kind of sequential data. In this architecture, for every input in a sequence a state is obtained and passed as additional information to process the following input of the sequence. The state of a particular input depends on the input itself but also on the state obtained from the previous inputs.

SketchRNN [30] is an example of RNN applied to sketches as sequences. This work proposes a generative network based on a sequence-to-sequence Variational Autoencoder [33][34] trained to generate or complete freehand sketches that is also capable to map human sketches to an embedding space where meaningful interpolation is possible. SketchEmbNet [35] proposes a similar Autoencoder architecture, but instead of being sequence-to-sequence it is image-to-sequence, with a CNN as encoder and a RNN as decoder. It exhibits interesting properties of the learned embedding space of images, showing how imitating drawing is a promising direction to learn general visual representations.

The Transformer [28] is a more recent sequence-oriented architecture, that had an important impact on the research community, with thousands of already published papers, boosting performance on different tasks mainly in the field of Natural Language Processing (NLP). The key concept is the self-attention mechanism, that can be implemented in different ways and where the common idea is to represent each input token based on the degree of correlation with reference to the other ones in the sequence, thereby better capturing the long-range dependencies between the data.

SketchFormer [36] implements a Transformer based encoder-decoder architecture that learns to embed input sketches in vector form, i.e. sequences of strokes. The proposed solution surpasses previous methods in term of sketch recognition and sketch-based image retrieval, showing also how an attention-based method is able to better manage more complex sketches with longer sequences when it comes to sketch generation and interpolation. Pixelor [37], instead, proposes a drawing agent based on a Wasserstein autoencoder [38] with a Transformer encoder capable to compete with humans in a Pixionary-like game. The game consists in drawing a sketch of a given subject to make it recognizable in the least amount of time possible. The focus is on identifying the key strokes that make a sketch recognizable. Pixelor, even after humans are trained to improve in the game, is still able to perform on par to human players. In [39] both presented representations

are used. A sketch specific self supervised approach is proposed where a sketch is transformed from a raster to a sequential representation and vice versa to learn more meaningful features from sketches. This work makes use of CNN like ResNet [40] and sequence-oriented networks like RNN and Tranformers.

Finally, it is also possible to represent a sketch as a graph. Considering a sequence of strokes, a single node of a sketch graph can be defined as the point of a stroke and an edge as connecting two nodes by proximity in the sequence. For example, in [41] are defined a local graph, where the proximity in the sequence is considered only for nodes that belong to the same stroke, and a global graph, where instead the edges are defined on temporal proximity and only for nodes belonging to different strokes.

Graph Neural Network (GNN) [42] is a type of architecture that aims to extend neural networks to non euclidean problems like graphs and manifolds. Graph Convolutional Network (GCN) [43] is an example of GNN, where, considering a layer of the network, every node has a state and for every node all the neighbors (connected nodes) states are processed to update the state of the considered node. Then, all the nodes have an updated state and the same procedure take place for all the subsequent layers.

Different works have been published combining GNNs and the sketch graph representation to tackle diverse tasks [41][44][45]. [41] is the first work to represent a sketch as a graph and proposes a new type of GNN, the Multi Graph Transformer (MGT). This work achieves similar retrieval performances compared to CNN alternatives while it has a lower inference time. SketchGNN [44] follows this line of research, proposing a GNN for semantic segmentation of sketches, improving over previous methods and having lower parameters compared to raster image and sequential solutions.

2.3 2D Sketch-based Tasks

In the literature many different sketch-based tasks have been considered over time. In the following some of the fundamental sketch-based tasks are introduced.

Sketch recognition [46][47][48][49][41] concerns the identification of the type of object or scene depicted in a sketch, normally defining a certain amount of different possible classes. Different approaches has been proposed to achieve sketch recognition. For example, learning a shared embedding structure between sketches and images [46], taking advantage of Transformers and the attention mechanism [41][47], defining sketch specific augmentation techniques [48] and proposing a sketch representation that counts for freehand sketch abstract nature [49].

Retrieving different kind of data from sketches, like **images** [50][24][51][52][53][1] or **videos** [54][55][56][2], is another very popular and important task. It takes advantage of the unique characteristics of sketching proposing it as an alternative to text-based retrieval. Drawing a sketch is

simple, does not require any drawing skill, nowadays one can draw with any mobile device and a sketch conveys information that may be hard to express by a textual description.

Many works have been proposed in the sketch-based image retrieval (SBIR) literature. "Sketch me that shoe" [50] proposes two datasets with {image, sketch} pairs regarding shoes and chairs and a triplet ranking model. Sketchy Database [24] introduces the first large scale dataset of {image, sketch} pairs with 125 categories and a model that embeds images and sketches in the same latent space. In SketchHelper [51], a neural network assists the user while drawing, improving the sketch quality and, as a result, the retrieval performance. [52], instead, proposes a self-supervised pre-training strategy based on solving Jigsaw puzzles. Recently, in StyleMeUp [53] a cross modal VAE [33] with a meta-learning strategy accounts for different user styles to improve generalization over never seen user sketches. [1] mitigates the {image, sketch} pairs data-scarsity problem proposing a semi-supervised technique that takes advantage from the image availability without the use of additional sketches.

Concerning video retrieval instead, [54] is the first to address sketch-based video retrieval, using queries composed of hand drawn objects along with their movements depicted as lines, curves and arrows. In [55], video content is summarized by clustering visual key-points trajectories and matching them with information extracted from query sketches. This method is extended in [56], where the search process is based on queries made out of both sketches and text. More recently, [2] addresses video retrieval by processing target videos and query sketches in two separated streams, one for appearance and one for motion.

Generative sketch-based tasks have also been addressed. Two common tasks are learning how to automatically draw sketches and generating realistic images from sketches. Investigating the art of drawing through sketch generation serves multiple purposes, from a better understanding about how we, as humans, are able to abstract and draw sketches, potentially also learning powerful general visual representation [35], to assist people while drawing [30], achieving collaborative drawing between humans and AI and developing games [30][37].

Lately, DoodlerGAN [57] proposes two new creative sketch datasets, depicting creative birds and creatures, and a part-based Generative Adversarial Network (GAN) [58] able to generate creative drawings. DoodleFormer [59], subsequently, improves creative sketch generation dividing the problem in two parts, localizing and sketching parts using graph-based Transformers (GAT) and convolutional networks. SketchGen [60], instead, tackles the CAD sketch generation problem proposing a generative transformer-based model and carefully designing a sequential language to manage the heterogeneity of primitives and constraints.

Generating images from sketches [15][61][62] can be useful to creative and artistic applications. It is possible to create art from simple sketches with any drawing skill level or aid artists in early phases of a project, inspiring them with a variety of images resembling a sketch.

Recently, SketchyGAN [61] proposes a GAN method able to generate images of 50 different classes of sketches, introducing a new data augmentation technique and using input images at multiple scales. [63], instead, is the first work to study unsupervised sketch to image synthesis, using unpaired data. Their main contribution is to split the task in two phases: geometry or shape synthesis, going from a sketch to a gray scale image, and enrichment, adding texture and colours.

In "Sketch your own GAN" [4], a pre-trained GAN is fine-tuned using one or a few user sketches of a subject, e.g. a cat, obtaining a GAN able to produce diverse images that resemble the target sketches. [3] considers the open domain sketch to image synthesis task, where images are generated also for sketch classes not available at training time. This work tackles both sketch to image and image to sketch tasks to take advantage of generated sketches to train on classes with missing data. Other works address tasks like stylized art synthesis [64], face image generation [65][66] or scene image generation [62][67].

While most of the works consider single object sketches, also some papers dealing with full scene sketches, considering foreground and background, has been published. Recently, Sketchy-COCO [62] proposes a new scene sketch dataset combining pre-existent real images and freehand sketches of individual objects, introducing a solution where real images are generated from scene sketches. SceneSketcher, instead, [68] considers image retrieval with scene sketches taking advantage of the dataset introduced in [62]. FS-COCO [69] proposes a dataset of newly collected freehand scene sketches and text labels paired with pre-existent real images and a method for image retrieval from sketches and text labels.

2.4 3D Sketch-based Tasks

The sketch literature comprehends a long investigation on the use of sketches to **retrieve** [70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 6] **and reconstruct** [87, 88, 89, 11, 90, 10, 12, 7, 91, 8, 92] **3D shapes**. Sketch based shape retrieval (SBSR), as already anticipated, concerns the retrieval of already available 3D shapes starting from a simple drawing conveying details of an object that are hard to describe with words. Being able to easily find already available 3D shapes may reduce the need to obtain new 3D shapes, a task that still requires a good amount of time and professional skills.

Among recent works, [84] proposes a view selection strategy to get the viewpoints that best represent a 3D shape and a cross-domain similarity model improving performances on SHREC datasets [73]. In [85], the authors focus their attention on the cost of supervised data, introducing a self-supervised solution with a modified triplet loss that reaches performance on pair with state-of-the-art. [6] tackles the zero-shot sketch-based 3D shape retrieval problem proposing a domain disentangled generative adversarial network (DD-GAN) that transforms the zero-shot task in a supervised task generating the missing sketches.

3D shape reconstruction from sketches aims to facilitate the modeling process by reducing time and skills required to obtain the desired 3D shapes. An important work in the literature is Multi-view Convolutional Networks [89], that considers a maximum of three sketches of the same object (front, side and top views) and leverages multi view depth and normal maps prediction to fuse them into a 3D point cloud by solving an optimization problem. A different work, [11], proposes to synthesize training sketches simulating freehand human style and to normalize them before directly predicting the corresponding 3D shape. [90], instead, introduces a sketch to 3D shape interactive tool where it is possible to create a good 3D model of a monster and define simple

animations for it. More recently, Sketch2Model [7] investigates the role of the sketch point of view in 3D shape reconstruction, showing how knowing the sketch viewpoint helps to interpret the sketch and leads to better reconstructions. Free2CAD [91] proposes a sequence-to-sequence model where a sketch is segmented into groups of strokes representing portions of the shape that can be produced with CAD commands.

Chapter 3

Inferring Depth from Scene Sketches

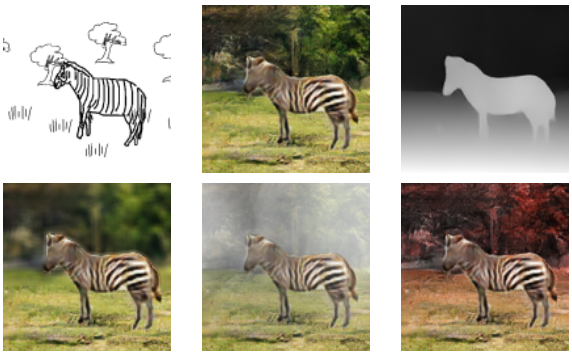


FIGURE 3.1: SketchyDepth generates images and depth maps from sketches (first row), enabling depth based creative effects (second row).

3.1 Introduction

In this chapter we discuss how the generation of 3D information from a scene sketch can improve 2D sketch-based tasks.

In literature there exist papers concerned with synthesizing 3D information from a sketch, but these works are mainly focused on single object sketches [88][89][11]. We instead focus on sketches depicting a scene, with an environment and potentially multiple objects, where the 3D information can play an even more important role. The geometry of a scene adds useful information about how the objects are located in the scene and how they are related, helping to correctly understand the scene itself. This additional information can be used to improve many different tasks like classification, segmentation, retrieval and image generation.

Following the recent publication of [62], we decide to work on the scene sketch to image generation task, proposing the first approach to generate geometrical information from scene sketches. In particular, we define a deep learning method that can synthesize both an RGB image as well as its associated depth map conditioned on an input scene sketch. We empirically demonstrate how leveraging geometrical information helps to generate higher quality RGB images and we also show the potential of leveraging depth maps in creative applications in two different ways. First, it is possible to manipulating the generated images by, e.g. Bokeh effect, fog, alteration of the illumination, 3D photos and others, as displayed in Fig. 3.1. Second, our framework enables depth

map sketching, a novel method to condition image generation by directly manipulating depth maps so as to refine the background content without requiring further training or supervision.

3.2 Related Works

3.2.1 GANs

The Generative Adversarial Network (GAN) is a type of implicit generative network first introduced by Goodfellow et al. [58]. This solution is composed of a generator and a discriminator that compete in a game with asymmetrical objectives. Given a dataset of examples (images in [58]), the generator is trained to output new examples that are indistinguishable from the dataset ones, i.e. that come from the same distribution. The discriminator learns to distinguish between real and fake examples. The generator objective is to fool the discriminator, generating fake examples that the discriminator identify as real, while the discriminator objective is to analyze real and generated fake examples and being able to properly identify them. The training procedure is based on an equilibrium where the generator and discriminator are both updated simultaneously and none of them overwhelm the other in the game. In the following we report some of the subsequent many different contributions on this line of research: architecture improvements [93, 94, 95, 96], conditional generation [97], image-to-image translation with cycle-consistency [98], convergence and alternative losses [99, 100, 101], high resolution [102] and GAN inversion [103, 104].

We propose SketchyDepth, a generative solution based on a conditional GAN architecture [97], that takes as input sketches as condition and outputs an RGB image and his corresponding depth map.

3.2.2 Scene-level Datasets and Tasks

SketchyScene [105] proposes the first dataset designed for scene level sketch understanding. This dataset is composed of 7264 human scene templates, 29056 scene sketches (considering augmentation) with semantic-level and instance-level annotation, 11316 object sketches (44 categories) and 4730 reference cartoon style images. Human experts use a tool to create scene sketches following reference cartoon style images and combining provided object sketches, that are characterized by different styles (different stroke width, abstraction, etc.). Subsequently, SketchyCOCO [62], proposed a new scene sketch dataset with natural reference images. A scene sketch contains foreground sketches that indicates specific object characteristics and background sketches that roughly indicate background composition. The dataset collects non-professional free hand object sketches from different datasets [24, 30, 16] and scene natural images from COCO stuff [106]. Using the segmentation labels of natarual images, scene sketches are automatically populated with background and foreground sketches. The dataset contains a total of 14081 scene sketches coupled with natural images.

Recent advances in the scene sketch research [15] have focused on image generation [62] and image retrieval [68].

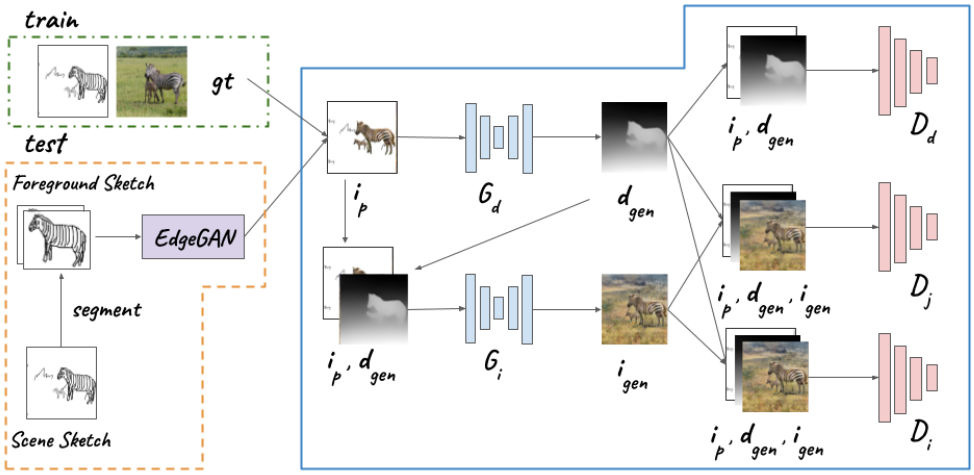


FIGURE 3.2: Overview of the SketchyDepth framework.

In SketchyCOCO [62] a proposed EdgeGAN generates foreground objects (cows, zebras, etc.) one by one following sketch details while a conditional generative network (`pix2pix`) [97] produces the final image, following loosely background indications (grass, sky, forest). SceneSketcher [68] proposes instead a fine-grained image retrieval solution from scene sketches exploiting graph convolutional networks [43].

We investigate further along the pathway of scene sketch research. We propose the first approach to generate geometrical and appearance information, i.e. an RGB-D image, from a scene sketch and we show how the generation of the geometric information improves the RGB generation and give access to creative applications.

3.3 Method

The proposed framework, dubbed SketchyDepth and illustrated in Fig. 3.2, takes a scene sketch as input and generates the corresponding RGB image and geometric information, the latter represented as a depth map. Akin to SketchyCOCO [62], the objects in the sketch are subdivided into foreground and background, where foreground sketches express precisely user intentions, i.e. scale, position and details of an object, while background sketches roughly indicate how the rest of the image should be completed, e.g. grass or trees in certain regions.

To train the framework we start from a dataset consisting of {*scene_sketch*, *scene_image*} pairs. As every object in a *scene_sketch* can be localized and classified by a sketch segmentation method [107], it is possible to crop the regions corresponding to foreground objects from each *scene_image*. Pasting these regions onto the corresponding *scene_sketch* yields *partial_images*, denoted as i_p in Fig. 3.2. Then, we complement the dataset with the *depth_map* associated with each *scene_image*, which is obtained by an off-the-shelf monocular depth estimation model trained by the authors on a mix of diverse datasets [108]. Thus, to train our generative model that returns the final image, i_{gen} and its associated depth map d_{gen} we rely on a dataset consisting of {*partial_image*, *depth_map*, *scene_image*} triplets.

As depicted within the orange dashed line in Fig. 3.2, the partial images required by our solution at inference time are obtained by applying the EdgeGAN model [62] to the foreground

objects detected by a sketch segmentation method [107]. EdgeGAN is trained separately based on a dataset consisting of {*object_sketch*, *object_image*} pairs, as described in [62].

3.3.1 Depth Map and Final Image Generation

To produce the depth map and final image corresponding to the input scene sketch, we design the conditional generative adversarial network [58][97] shown within the solid blue line in Fig. 3.2. Having obtained a partial image, i_p, for each *scene_sketch* in the training dataset as explained above, to train our generative model we create triplets $\{i_p, d_{real}, i_{real}\}$, where d_{real} and i_{real} are the *depth_map* and *scene_image* corresponding to each *scene_sketch*. Then, a depth generator, G_d, takes as input a partial image, i_p, and produces a corresponding depth map, d_{gen}:

$$d_{gen} = G_d(i_p) \tag{3.1}$$

To guide the generator a depth discriminator, D_d, is defined. D_d is trained to discriminate between fake $\{i_p, d_{gen}\}$ pairs and real ones, $\{i_p, d_{real}\}$. We leverage a conditional GAN loss [97] to train the generator G_d and the discriminator D_d:

$$\begin{aligned} \mathcal{L}_{gd} = \mathbb{E}_{i_p, d_{real}}[\log D_d(i_p, d_{real})] + \\ \mathbb{E}_{i_p}[\log(1 - D_d(i_p, d_{gen})] \end{aligned} \tag{3.2}$$

where $\mathbb{E}_x[\cdot]$ denotes expectation over x and the discriminator D_d is trained to maximize $\mathcal{L}_{gd}$, while the generator G_d is trained to minimize $\mathcal{L}_{gd}$. Similarly to `pix2pix` [97], the previous loss is complemented by an L1 term that encourages G_d to generate depth maps, d_{gen}, that are similar to the corresponding real ones, d_{real}:

$$\mathcal{L}_{rd} = \mathbb{E}_{i_p, d_{real}}[\|d_{real} - d_{gen}\|_1] \tag{3.3}$$

The objective for G_d and D_d is thus given by:

$$\mathcal{L}_d = \arg \min_{G_d} \max_{D_d} \mathcal{L}_{gd} + \alpha \mathcal{L}_{rd} \tag{3.4}$$

where α is a scale factor that weighs the contribution of the L1 term.

After depth generation, an image generator, G_i, takes as input the partial image, i_p, and the generated depth map, d_{gen}, in order to generate the final scene image, i_{gen}:

$$i_{gen} = G_i(i_p, d_{gen}) \tag{3.5}$$

An image discriminator, D_i, is defined as the critic of G_i. D_i is trained to discriminate between fake triplets, $\{i_p, d_{gen}, i_{gen}\}$, and real ones, $\{i_p, d_{real}, i_{real}\}$. Akin to depth generation, a conditional

generative loss combined with an L1 term defines the objective for G_i and D_i:

$$\begin{aligned}\mathcal{L}_{gi} =& \mathbb{E}_{i_p, d_{real}, i_{real}}[\log D_i(i_p, d_{real}, i_{real})] + \\ & \mathbb{E}_{i_p}[\log(1 - D_i(i_p, d_{gen}, i_{gen})] \end{aligned} \tag{3.6}$$

$$\mathcal{L}_{ri} = \mathbb{E}_{i_p, i_{real}}[\|i_{real} - i_{gen}\|_1] \tag{3.7}$$

$$\mathcal{L}_i = \arg\min_{G_i} \max_{D_i} \mathcal{L}_{gi} + \alpha \mathcal{L}_{ri} \tag{3.8}$$

Furthermore, to encourage alignment between d_{gen} and i_{gen}, we train a joint discriminator, D_j, to tell apart fake $\{i_p, d_{gen}, i_{gen}\}$ triplets and real ones, $\{i_p, d_{real}, i_{real}\}$. The objective for D_j is defined as a conditional generative loss where both generators are optimized:

$$\begin{aligned}\mathcal{L}_{gj} =& \mathbb{E}_{i_p, d_{real}, i_{real}}[\log D_j(i_p, d_{real}, i_{real})] + \\ & \mathbb{E}_{i_p}[\log(1 - D_j(i_p, d_{gen}, i_{gen})] \end{aligned} \tag{3.9}$$

$$\mathcal{L}_j = \arg\min_{G_i, G_d} \max_{D_j} \mathcal{L}_{gj} \tag{3.10}$$

The complete training procedure alternates the update of every discriminator with an update of both generators with respect to $\mathcal{L}_i$, $\mathcal{L}_d$ and $\mathcal{L}_j$.

3.4 Experiments

3.4.1 Dataset

In the experiments we use the SketchyCOCO dataset [62], a dataset consisting of {*scene_sketch*, *scene_image*} pairs that contains 14 foreground classes (like horse, giraffe or elephant) and 3 background classes (trees, grass and sky). Indeed, although image generation from sketches is a generative task, and therefore open-ended, the dataset features for each sketch a "ground-truth" real image which was paired by the authors with the sketch and can be used as an example of a valid result.

To evaluate the proposed conditional generative network, we start from pairs {*partial_image*, *scene_image*}. Partial images are obtained as detailed in Section 3.3. These are the same pairs used in SketchyCOCO to train and test the background generation. We then add depth supervision for ground-truth images (and, in turn, the associated sketch) by running a pre-trained version of MiDaS [108], a popular monocular depth estimator trained for generalization. The final datasets contain 11265 training triplets and 2816 test triplets. To validate our experiments we used a split of the training dataset with 9576 training triplets (85%) and 1689 validation triplets (15%).

3.4.2 Baseline and Implementation

As baseline for scene image generation we use the SketchyCOCO proposal [62]. With this method, in the background generation phase a `pix2pix` network [97] is trained to learn to transform foreground images (i.e. partial images) into scene images. The configuration for the training and

Method	FID ↓	SSIM ↑
SketchyCOCO (original) [62]	164.80	**0.2880**
SketchyCOCO (replicated)	130.48	0.2766
SketchyDepth (w/o G_d and D_d)	130.38	0.2820
SketchyDepth (w/o D_j)	136.13	0.2783
SketchyDepth	**116.87**	0.2693

TABLE 3.1: Quantitative scene image generation results. We report FID (lower is better) as image quality metric and SSIM (higher is better) as faithfulness metric among generated images and ground-truth images.

evaluation differ from the `pix2pix` default ones as follows. A pre-processing operation is added so as to crop images centrally to obtain the largest squared image and then resize it to 128×128; the `pix2pix` resize and crop pre-processing operations are set to size 128×128 so that they do not modify the images further; as for the generator, the UNet128 [109] architecture proposed in `pix2pix` is selected, the output size of the generated images is 128×128, and the system is trained for 110 epochs. All the other hyper-parameters are kept aligned with the `pix2pix` code.

We trained the baseline to replicate SketchyCOCO results but we were not able to exactly replicate them, even if we used the same official code for `pix2pix` and we had the kind support of the authors in several private communications. This is probably due to different versions of the official `pix2pix` pytorch code[1] [97] [98] that got many fixes and updates over time. Since we were not able to recover the exact version of `pix2pix` code used in SketchyCOCO, we selected a recent version of `pix2pix` pytorch code to have an implementation of reference for all the experiments[2]. Thus, in the experiments reported in Table 3.1 and discussed in the next subsection, we report the SketchyCOCO paper results (first row) as well as the results we have replicated by using the selected `pix2pix` version (second row), which we consider the baseline.

We used the same PyTorch [110] implementation for all the operations in common between the baseline and our proposal, like pre-processing and data augmentation. In our adversarial training, we used `pix2pix` default PatchGAN [97] architecture as discriminators. The depth map follows the same pre-processing of the partial images but processed as single channel. The α L1 balancing term is equal to 100 and we use the Adam optimizer [111] with 0.0002 as learning rate, as done for the baseline. All other hyper-parameters share the same value in the baseline and our method. We used a 1080ti and a 1060 nvidia gpus for our experiments.

3.4.3 Scene Image Evaluation

We evaluate the images generated from the sketches belonging to the test set by comparing them with the corresponding ground-truth images according to Fréchet Inception Distance (FID) [112] and SSIM [113] metrics.

FID is a measure of similarity between two sets of images that correlates well with human judgement of visual quality and it is often used to evaluate the realism of generated images. Thus, we use it to compare the set of generated images and the set of ground-truth images in order to evaluate the image generation quality.

[1] github: https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix
[2] commit: f13aab8148bd5f15b9eb47b690496df8dadbab0c

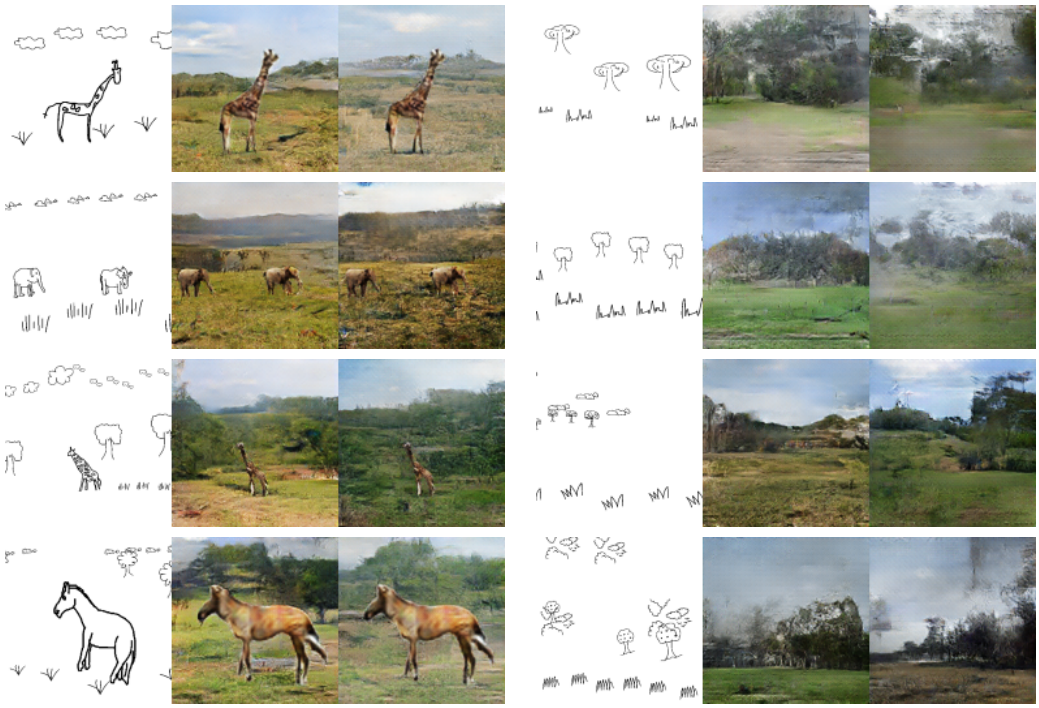


FIGURE 3.3: Scene image generation from sketch. Results are visualized in two columns: each column shows from left to right the input sketch, our result, and the baseline. The first column deal with sketches featuring foreground and background content while in the second the sketches contain only background classes.

The SSIM metric measures the similarity between a pair of images and it is normally used as a quality measure when the perfect image one wishes to attain is known (e.g. in super resolution applications). In the context of scene generation from sketches, the SSIM has been proposed as a measure of faithfulness to the sketch of the generated image by using the ground-truth image as a proxy for the sketch. However, we argue the SSIM to be less meaningful than the FID for a generative task like scene from sketches, because the "ground-truth" image is just one of the possible high-quality results and not the only correct one, and slightly higher SSIM with respect to it may not really measure a lower faithfulness to the sketch, especially for background regions which are coarsely indicated in the sketch and may instead vary wildly between generated images. Nonetheless, we report SSIM scores as done in [62].

Table 3.1 collects the experimental results. The first row reports the original SketchyCOCO paper results, while the second row those obtained trying to replicate it, which we use as baseline to compare the results provided by our framework, shown in row 5.

Our framework yields better FID with respect to the baseline, this highlighting that the images generated by our method are usually of higher quality with respect to those generated by the baseline. The SSIM score, instead, is slightly worse: as discussed above, we believe this result mainly indicates that the content generated by our method for a specific sketch differs more from the corresponding ground-truth image. Yet, it may still represent well the input sketch, as shown by the qualitative results in Fig. 3.3.

As reported in Table 3.1, beside testing the SketchyCOCO baseline and our proposal, we also consider two additional configurations as an ablation study. In particular, on one hand, we wish

to sift out the actual contribution of the joint depth map generation to the image generation performance of our framework, while on the other we wish to validate the impact of the joint discriminator D_j.

Removing all components strictly needed to produce and discriminate the depth map in our proposal, i.e. G_d and D_d, we are left with the generator G_i and the discriminators D_i and D_j, modified so as to not take a depth map as input. This can also be thought of as the baseline with two discriminators, i.e. with the additional training signal of D_j for G_i. Row 3 shows the results of this experiment, while row 4 ablates the contribution of the joint discriminator D_j.

Ablating out the joint depth map generation, SketchyDepth obtains the same FID score as the baseline, showing how the additional training signal for the image generator, considering only the RGB image information, does not help to reach higher image quality generation. This validates that the FID gain of our method is related to the introduction of depth information, and shows how depth information can impact significantly the sketch to image task, in particular increasing image generation quality. Interestingly, the SSIM of this configuration increases with respect to the baseline: more supervision for the generator within the same budget for training time forces it to produce images more similar to the ground-truth ones, which increases SSIM but does not increases realism as measured by the FID, another hint of the biased nature of the SSIM and the limited relevance of small differences in its value to assess the performance of the sketch to scene image generation task.

Similarly, our solution without D_j cannot improve performances over the baseline. This result show how, beside introducing the additional depth information as studied in the previous experiment, it is also important to have the correct amount of training signal and alignment between d_{gen} and i_{gen}, as introduced by D_j, to be able to improve image quality generation. This validates that all components of our framework are needed to deliver improved performance with respect to the baseline in the sketch to scene image generation task.

In Fig. 3.3, we show images generated from test sketches. The first column concerns sketches with foreground and background content, the second one sketches with background content only. Visual observation suggests that depth information contributes to fill effectively the empty regions of the input sketch, see in particular column 2 rows 3 and 4, upper right portion of images. We further investigated considering the FID values subdividing the test examples in foreground-background (fb) and background-only (bo), obtaining 96.70 in fb and 184.73 in bo for the baseline and 92.91 in fb and 162.24 in bo for the proposed method. This quantitative measurements seem to support our observation concerning the ability of our solution to contribute more in terms of background generation.

3.4.4 Depth Evaluation

There are no previous methods that can generate a depth map from a scene sketch. However, it is possible to obtain depth maps from images by running depth-from-mono models. Thus, to asses the quality of the depth maps generated by our method, we compare them with those generated by a pre-trained monocular depth estimator, i.e. MiDaS [108], applied to the generated RGB images. We apply MiDaS on generated images because real RGB images are not available at test time. Furthermore, even if they were available, the comparison would not be meaningful

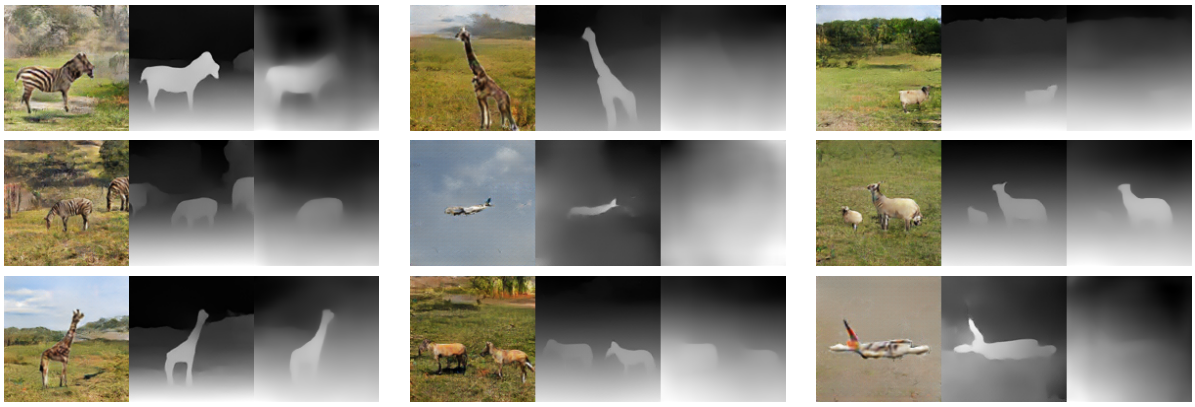


FIGURE 3.4: Depth map generation from sketch. Results are visualized in three columns. We show from left to right: our generated scene images, our generated depth maps and depth maps generated by MiDaS from our generated images.

because, for the same sketch, the real image and the generated one exhibit different structures and therefore different depth maps.

Fig. 3.4 shows examples of depth map generation. In each of the three columns we show, from left to right, generated images, the corresponding depth maps generated jointly by our method and the depth maps obtained by MiDaS on generated images. Our method produces good depth maps most of the times, while MiDaS depth maps are often less detailed, in particular on foreground objects, and sometimes grossly wrong. This is somewhat surprising since MiDaS is also used to create depth supervision at training time for our framework, and it shows how monocular depth estimation algorithms are influenced by subtle changes in the image texture [114], which occurs in our generated images. This also vouches for the importance of simultaneous generation of RGB and depth information, as proposed in our framework, to avoid such depth artifacts or errors.

To further assess our depth map generation results we conducted a human evaluation. We selected 100 random sketches containing foreground and background from the test dataset and generated the corresponding image and depth map with our method. From the generated images we also obtained depth maps by MiDaS. Given the generated image and both depth maps, the test proposed to the participant consists in choosing which of the depth maps best corresponds to the displayed image. We collected data from 23 people with previous experience with computer vision and depth maps. Over $23 \times 100 = 2300$ answers the 23 participants preferred our generated depth maps compared to those yielded by MiDaS 2153 times, i.e. about **93.60**% of the times. Human evaluation results are therefore aligned with the previous qualitative considerations.

3.4.5 Depth Based Creative Applications

One of the interesting applications unlocked by generating depth maps together with images is the possibility to apply effects over images which vary according to depth values. Given an effect, it is possible to choose the depth value where the effect will start to affect the image, the direction of the effect, i.e. if the intensity of the effect increases or decreases alongside depth value, and the maximum intensity of the effect. We experimented with three different effects.

FIGURE 3.5: Leveraging depth map information to apply creative effects on generated images. In both columns from left to right we show generated images, bokeh effect, light variation, fog and hue shift.

FIGURE 3.6: 3D photo example: simulation of the movement of the camera toward the foreground of the scene, i.e. a zoom effect is simulated.

The first one is a blur filter whose intensity increases with the estimated depth of the central pixel, in particular the weight of the central pixel is scaled accordingly. This solution can be used to reproduce a shallow depth of field or a Bokeh effect[3].

Light, hue shifts, and colour to grayscale transitions, where the intensity of the effect increases along with depth, are also possible and can be realized transforming the image to the HSV color-space and scaling a specific channel of pixels according to depth value.

Finally, climatic fog and the transition-to-cartoon effects can be realized by using an overlay image of the desired effect, i.e. a fog image and a cartooned version of the input image. The output image is produced as a weighted average of the input image and the overlay image where the weight varies according to the depth value at every pixel. We note that in the literature also more sophisticated depth-based algorithms [115] to introduce simulated fog into images do exist, which could be beneficially leveraged starting from our generated depth map.

Fig. 3.5 reports examples of the above mentioned effects applied to images generated by our method based on the associated generated depth maps. In particular Bokeh, light variation, fog and hue shift effects are shown in the figure, while more effects, i.e. colour to grayscale and transition-to-cartoon, are visible in Appendix A.1.2.

We also experimented with the creation of so called 3D photos, i.e. a video where a 3D effect is applied. We used a pre-trained model [116] to produce 3D photos using our generated depth maps and images. Fig. 3.6 shows some sequences of frames sampled from the generated 3D photos.

Overall, these qualitative results show how obtaining geometrical information from a sketch enables different and engaging effects that exceed the limitations of the 2D image plane. The proposed effects are just an example of the possibilities unleashed by the availability of depth maps, and more creative uses can be explored.

3.4.6 Can You Sketch a Depth ?

The design of our pipeline enables the exploration of another creative use of depth maps. Once our system is trained, we can get the partial image of a sketch from a pre-trained EdgeGAN and use G_d to get the corresponding depth map. Since the image generator G_i is then conditioned on it, if we manually modify the depth map with an image editor before feeding it to G_i, we can gain additional control over image generation. For instance, we can add shapes of objects that belong to background classes or modify the structure of the image, e.g. the height of the horizon. In our experiments, we treat the depth map as a grayscale image and edit for every new object/area a single fixed grayscale value, which defines its depth. After the depth sketching phase, we can use the resulting depth map and the partial image as input to G_i to obtain the corresponding generated image.

Depth sketching examples are visible in Fig. 3.7. Qualitative results show how our system can generate new objects and partially modify the image structure when conditioned on a sketched version of the generated depth map, without further training or supervision. The system learns to do so only from the edited depth shapes, in particular G_i gains knowledge about the shapes

[3] https://en.wikipedia.org/wiki/Bokeh

FIGURE 3.7: Sketching generated depth maps to acquire finer control over image generation. In both columns, from left to right: generated depth map, generated image, sketched depth map and newly generated image.

of objects and image structures that occur often in the training dataset. Depth sketching gives us more control over position, shape, and scale of objects which are normally part of the background. Indeed, this information cannot be specified directly in the sketch, where background classes are meant as coarse selectors of the background texture, not as precise indicators of the appearance of the objects forming the background. As in our setting the background objects that appear more frequently are grass, sky and trees, we can get additional and finer control mainly over different types of trees, bushes, vegetation.

Although depth sketching is an interesting additional way to gain control over the generated image, it is enabled by our framework almost as a side effect and it comes with its own limitations: in particular, sometimes more than one attempt is needed to obtain the desired result. We believe that a training protocol specifically designed with this use case in mind, e.g. with data augmentation simulating the test conditions, could easily enhance its robustness.

3.5 Conclusion

In this chapter, with the presented framework, it was possible to demonstrate how geometric information can be generated starting from a scene sketch and how it helps to improve a 2D sketch-based task.

In particular, SketchyDepth is the first framework to generate a depth map and an image from a scene sketch. We demonstrate how leveraging geometrical information allows for improving scene sketch to image generation quality and how the presented framework can generate depth maps that are consistently better compared to an alternative method.

Moreover, generated depth maps can be used to obtain many depth-based effects over generated images which offer a variety of tuning nobs to creative users. An introduced creative depth

manipulation technique is Depth Sketching, where sketching the generated depth maps gives finer control over the generated background.

We hope that our findings may foster further investigations dealing with depth maps generation. For instance, one might conjecture that leveraging depth information could improve the results in generative tasks beyond sketches, e.g. in a classic RGB generation task with state of the art generative models.

Chapter 4

Investigating Vision-Language Models for Sketch-Based 3D Tasks

4.1 Introduction

In this chapter we investigate the use of Vision-Language models to overcome the limits of sketch-based 3D tasks. In particular, we study Vision-Language models to overcome the domain gap between freehand sketches and other kind of image domains, like line or RGB renderings, without the need of a training phase on {freehand sketch, 3D shape} pairs, that are limited and very costly to acquire.

This chapter is subdivided into two parts. In the first one, we conduct a study that investigates the embedding space of Vision-Language models when different sketch related domains are considered. We then apply Vision-Language models to sketch-based 3D shape retrieval proposing a solution that achieves state-of-the-art performance without using supervised data for training. In the second part, we extend the investigation to the 3D reconstruction from sketch task. We experiment with an inversion framework based on a 3D generative network, showing how a CLIP perceptual loss is able to transfer features from objects depicted in freehand sketches to the inverted 3D shapes, a promising research direction whose future extension is discussed at the end of the chapter.

4.2 CLIP-facilitated sketch-based 3D shape retrieval

We focus on leveraging large Vision-Language pre-trained models for sketch-based 3D shape retrieval without modifying their original feature spaces. Fine-tuning large models is likely to result in the loss of certain properties, instead we propose a framework and methodology for finding the best settings for stronger retrieval performance.

We first conduct a study on the use of a large Vision-Language model, CLIP[13], to obtain a perceptual loss, analyzing the properties of the embedding space when applied to sketch related domains. We present the study in section 4.2.3.1, where we analyze the ability of CLIP to differentiate between various objects and different views. In this study, we consider three different domains: freehand sketches, line renderings and RGB renderings.

Motivated from the study results, we tackle the problem of sketch-based 3D shape retrieval. Following previous work on sketch-based 3D shape retrieval, we represent 3D shapes using their

multi-view projections [78, 80, 82, 83, 86, 72, 75, 84]. We also assume that for each class of 3D shapes, there are several viewpoints, from which humans are more likely to depict an object [72, 75, 84, 5, 23]. Our methodology consists of careful exploration of individual factors that can affect retrieval performance: (1) we study how the selection of viewpoints for 3D shape representation affects retrieval accuracy, assuming that we know a sparse set of most likely viewpoints for each shape class; (2) we study the robustness of features under affine transformations of objects in views; (3) we study the augmentation of query and target shape views with affine transformations to increase the robustness of the computed features; (4) we evaluate the domain gap between freehand sketches and view renderings, comparing line and RGB renderings. As a result of these studies, we derive the best settings that allow to improve retrieval accuracy compared to a baseline by nearly 10 points.

Having in mind our goal of not modifying the original latent space, we observe that the feature space can contain many irrelevant features for a particular task. For instance, in our case when we compare freehand sketches and line renderings, the features representing color information, texture, etc., might harm the performance on the downstream task. We, therefore, propose a learning framework to mask or weight the latent space features. During training we rely on line-renderings only. However, we assume that a small set of paired freehand sketches and 3D shapes for at least one shape class is available to choose the best checkpoint. We then show that such model generalizes to freehand sketches of the same shape class and other shape classes.

Finally, we show that our approach outperforms previous supervised work in terms of top-1 retrieval accuracy by more than 17 points, while also slightly improving top-5 accuracy.

4.2.1 Related work

4.2.1.1 Sketch-based 3D shape retrieval

Category-level and fine-grained retrieval Most of works in sketch-based 3D model retrieval [70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 6] focus on the problem of *category level* retrieval: They aim to retrieve any instance of a particular object category. In other words, the retrieval is considered to be successful if, given a sketch of an object, the retrieved top N 3D models belong to the same category.

The only work we are aware of that studies fine-grained sketch-based 3D model retrieval in a supervised setting is the recent work by Qi et al. [5]. They collect a dataset of instance-level paired freehand sketches and 3D models, which we also use to test our model. They use triplet loss training [117], classic for retrieval tasks, and represent 3D models using multi-view RGB renderings. The main novelty of their paper lies in learning view attention vectors, which allow to reduce the domain gap between a sketch and multi-view projections of a 3D shape. We also address fine-grained retrieval but in contrast to this work, we rely on a pre-retrained CLIP model.

To reduce the domain gap between sketch queries and 3D models, several works [118, 119] study fine-grained retrieval from a 3D sketch created while wearing a virtual reality headset. Our model aims for much more accessible inputs that can be created with a computer mouse or on paper.

Multi-view feature aggregation Like the majority of the works on sketch-based 3D model retrieval, we use multi-view shape representation, however many of these works differ in how they aggregate features across viewpoints. Thus, Xie et al. [78] use the Wasserstein barycenter of 3D shapes projections in the CNN feature space to represent 3D shapes. He et al. [80] follow MVCNN [120] and aggregate views features with element-wise maximum operation across the views in the view-pooling layer. Lei et al. [82] proposed a representative view selection module that aims to merge redundant features for similar views. Chen et al. [83] learn multi-view feature scaling vectors which are applied prior to average pooling vector, in order to deal with non-aligned 3D shape collections. Qi et al. [5] learn view attention vectors conditioned on the input sketch, which allow to reduce the domain gap between a sketch and multi-view projections of a 3D shape. Zhao et al. [86] leverages spatial attention [121] to exploit view correlations for more discriminative shape representation. We explore here a simpler strategy where we represent a 3D shape with the view that is the most similar to the input query in the CLIP embedding space, that we also describe in Section 4.2.2.1.

Viewpoints selection While the works discussed in the section above represent 3D shapes using densely equidistantly sampled viewpoints and focus on how to combine multi-view features, several works [72, 75, 84] compute a set of best viewpoints for 3D shapes before models training. These methods rely on the assumption that there are human preferred viewpoints for sketching for each shape class and aim to identify them given a dataset with the class-level pairing between freehand sketches and 3D models. These works first represent 3D shapes with dense multi-view line renderings. To compute the similarity between freehand sketches and line renderings, Lu et al. [72] rely on SHOG feature histograms [70]. They then compute for each class and for each 3D model view the average similarity to the sketches in the dataset and select the N viewpoints with the highest similarity scores. Zhao et al. [75] extract strokes from bitmaps and compare their similarity in proximity to sketch key points obtained with the Harris corner detector [122]. They divide the viewpoints into positive and negative samples based on the average probability of sketches to be drawn from those viewpoints. They then use an SVM (Support Vector Machine) classifier to predict for a given shape class and for each shape viewpoint the probability of being drawn by humans. Finally, they follow the strategy of Lu et al. [72] and propose a penalizing term for selecting too similar viewpoints. Xu et al. [84] rely on a pre-trained model [123] to extract freehand sketch and line drawing features and follow Zhao et al. [75] to select the set of best viewpoints for each shape class. In our work, we do not assume that we have an access to sketches from a new shape class a priory. Therefore, we carefully study the robustness of our algorithm with respect to the choice of 3D shape viewpoints.

4.2.1.2 Multi-modal retrieval with CLIP

Multi-modal retrieval is not directly related to our work, but two concurrent works [124, 125] to ours are worth mentioning as they rely on CLIP. They explore CLIP embeddings for retrieval from multi-modal inputs such as 2D sketches or images and text. Sangkloy et al. [125] study image retrieval and focus on fine-tuning CLIP using triplets of synthetic sketches, images, and their captions. They rely on the availability of textual descriptions matching their images. In

our case, there is no dataset with textual descriptions of 3D shapes, therefore their approach is not applicable to the problem we study. Schlachte et al. [124] study 3D model retrieval. They represent 3D shapes with three RGB views: front, back, and 3/4 bird-eye viewpoint. We conduct a study of viewpoints selection and in Appendix B.4 we show that this is not an optimal viewpoint selection strategy. The main focus of their work is on exploring the weighted fusion of CLIP features from multiple inputs for artistic control. Instead, in our work, we focus on determining the optimal strategy for 3D shape retrieval from freehand sketch inputs using CLIP.

4.2.1.3 Sketch datasets

With the advent of sketch datasets [70, 72, 126, 24, 127, 23, 5, 128] the research on sketching thrives. However, it is costly and challenging to collect dataset of freehand sketches, especially when instance-level pairing between several domains is required. The common practice is to let participants study a reference image for a short period of time and then let them draw from memory [70, 126, 24, 128, 125]. This task becomes increasingly challenging when the required pairing is between 3D shapes and sketches, as one has to ensure that the viewpoints are representative of those that people are more likely to sketch from [72, 75, 84, 5, 23]. To the best of our knowledge, there is only one dataset [5] of freehand sketches paired with 3D shapes obtained by participants with no prior art experience, that takes views into account and follows the protocol of sketching from memory. The small dataset collected by Zhang et al. [7] contains only one sketch viewpoint for each object, and the viewpoints are uniformly sampled around 3D shapes. Some sampled viewpoints are not meaningful for retrieval and there are too few examples for performance evaluation. The recent dataset of paired sketches and 3D models of cars [12] has the same problem of viewpoint and moreover the sketches are drawn directly on top of image views and mostly contain outer shape contours. We, therefore, evaluate our approach on the dataset by Qi et al. [5], as it is the only existing representative dataset with instance-level pairing between sketches and 3D shapes.

4.2.2 Method

In this section, we report the sketch-based 3D shape retrieval methodology, while the conducted study on Vision-Language models is reported in the experiments (Section 4.2.3.1).

We start describing the optimal setting of the retrieval model and then we describe our learning approach to mask or weight the features of the pretrained CLIP[13] model to keep only those that are most relevant to sparse sketch comparisons.

4.2.2.1 Zero-shot

Since the CLIP model [13] is trained on pairs of textual descriptions and images, we represent a 3D shape with its multi-view projections. Multi-view shape representation was shown to give best results by several previous works [5, 85], as it allows to reduce the domain gap between 2D sketches and 3D shapes. In this project, we use synthetic sketch rendering, also commonly referred to as non-photorealistic or line rendering, instead of classical RGB renderings. We resize all sketches to the same size of 224×224. Empirically, we found that scaling a single object to

occupy the central area with the size of 129 × 129 pixels leads to the best performance. To obtain the list of ranked 3D shapes according to their similarity to the query sketch, we compute the embeddings of the sketch and each 3D shape view with CLIP. We compute the distance between the two embeddings as a Mean Squared Error (MSE). We then assign the distance between a sketch and a 3D shape as the minimum distance between a sketch embedding and views embeddings. Formally, this can be written as follows:

$$\mathrm{d}(\mathrm{Sk}, \mathrm{Sh}) = \min_{v \in views} \mathrm{MSE}(\mathrm{CLIP}(\mathrm{Sk}), \mathrm{CLIP}(\mathrm{v})), \tag{4.1}$$

where Sk and Sh stand for sketch and shape, respectively; and $CLIP(\cdot)$ denotes features extracted with the CLIP image encoder [13].

For a more robust retrieval behavior, we propose to compute the features of sketch views and line views by averaging the features of the original views and the views transformed with an affine transformation. In our work, we carefully study the optimal translation and rotation affine parameters for this purpose and show that their performances are consistent across different shape categories.

4.2.2.2 Learning relevant features

As CLIP [13] is a general model trained on millions of text and image pairs, its feature space might contain information about colors, texture, lighting, etc. Therefore, we study if we can learn to weight or mask certain features to improve the accuracy of matching line renderings and sketch views.

Our learning framework consists of a feature extractor, a modulating layer m and a contrastive objective. For simplicity, we use the CLIP contrastive loss [13], which is a simple cross entropy contrastive loss [129]. We represent all considered 3D shapes with N views as line renderings (We provide the exact implementation details in the results section). A batch of data is composed of two set of line renderings, where each set has line renderings of the same objects but different viewpoints. We treat object identity as a class label, effectively ensuring that the views of the same object are projected closer to each other than views of different objects.

We train only weight parameters of the modulating layer, while keeping the CLIP model weights intact. The modulating layer takes as input the CLIP line view embedding. Namely, given a line view embedding x with dimensions S, we define a modulating layer $m(\cdot)$ as the Hadamard product between trainable weights w_m, with the same dimensions S, and the input tensor:

$$m(x) = x \odot w_m. \tag{4.2}$$

We consider two types of the modulating layer: the one that learns to weight CLIP features, and the one that learns to mask non-relevant features. To achieve this, we let the weights of the modulating layer to be continuous or binary, as we detail below. The backward pass is the same for all the scenarios and constitutes the conventional gradient descent backpropagation pass. The forward pass differs in the way the weights are used by the modulation layer. We discuss the two settings that we consider below.

The *continuous* setup simply uses the continuous values of the weights w_m.

The *binary* version of the weights is defined as follows:

$$w_{m,binary} = \begin{cases} w = 1, & \text{if } w \geq \theta \\ w = 0, & \text{if } w < \theta \end{cases} \tag{4.3}$$

where w are individual components of w_m, and θ is the threshold hyperparameter (0.1 in our experiments).

4.2.3 Experiments & Results

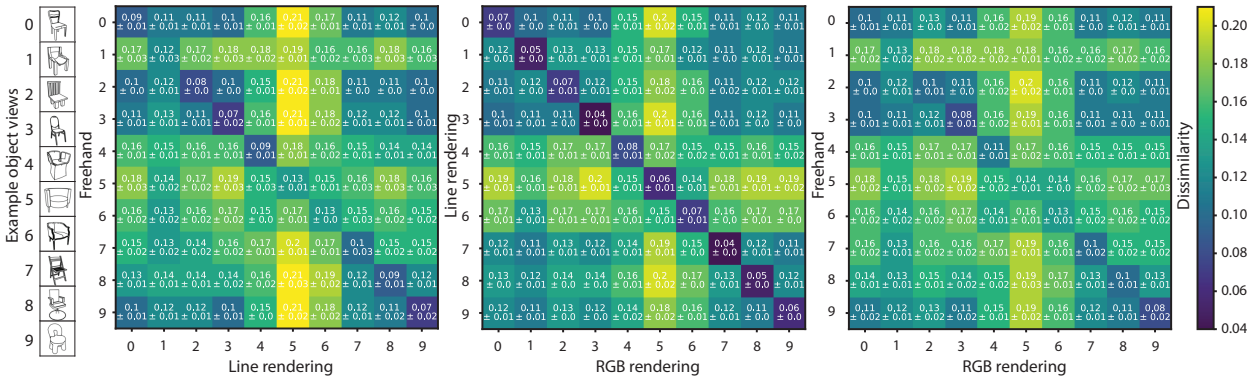


FIGURE 4.1: We plot pairwise distances between shapes, when their views come from one of the three image domains: freehand sketches [5], line or RGB renderings. To the left are shown the 10 random sampled examples. See Section 4.2.3.1 for the details.

Datasets We evaluate various strategies on the sketches from the existing dataset of freehand quick and abstract sketches paired with 3D shapes [5]. This dataset contains sketches for two shape categories: chair and lamp, that represent 3D shapes from the ShapeNet dataset [130]. The sketches are created by participants without any prior sketching experience, and fit well the scenario we are targeting. The sketches are drawn with a fixed zenith angle of around 20 degrees. For each category three settings of azimuth angles are used. For the chair category, they are 0°, 30° and 75°, while for the lamp category they are 0°, 45° and 90°. These particular viewpoints are selected based on sketching literature and pilot studies, as the most likely viewpoints. We refer the reader to the paper by Qi et al. [5] for a detailed discussion.

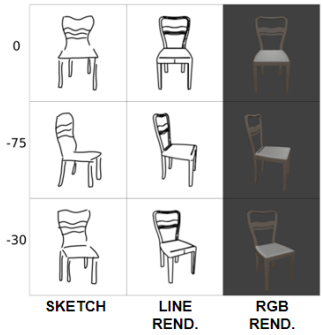
This dataset contains a split consisting of a training and a test set. We use a small set of 200 chair and 200 lamp sketches from the training split as a validation set for most of the experiments in this section. We chose this number to facilitate the comparison with performance on the test set. We then show that our results generalize to the test split.

Multi-view line rendering To obtain line renderings, we use Blender Freestyle[131]. We keep the original shape normalization from the ShapeNet dataset [130]. We render views using silhouettes and creases lines, with a uniform stroke width set to 2.2, we set the camera distance to an object to 2.5 and set the camera zenith angle to 20°.

4.2.3.1 How well can CLIP differentiate between views and objects?

We target retrieval from quick and abstract sketches, such as sketches collected by Qi et al. [5] and Zhang et al. [7]. There are several strategies to represent 3D shapes using their multi-view projections in sketch-based 3D model retrieval literature: using RGB renderings or line renderings. However, there is a large domain gap between such views and freehand sketches we consider as queries. To motivate the use of Visual-Language models, we study CLIP [13] features similarity across domains, and its ability to match the model views across various domains. In this section, we only consider 3D shapes from the chair category from the *AmateurSketch* dataset [5].

We use three different views for every object, with the camera azimuth angles set to 0°, 30° and 75°. The inset on the right shows example viewpoints for one of the considered 3D shapes. In this section, we use the third layer of the pre-trained ResNet101 CLIP image encoder as our CLIP embedding space, taking inspiration from [132].



Aggregated feature similarity We consider 10 randomly-selected objects (the corresponding freehand sketches for them are shown in Fig. 4.1), and compute the distance between two 3D shapes A and B, represented with their multi-view renderings or freehand sketches from the three considered viewpoints, as follows:

$$d(A^i, B^j) = \frac{1}{V} \sum_{k=1}^{V} \Big(\text{CLIP}(A^i_k) - \text{CLIP}(B^j_k) \Big)^2 \tag{4.4}$$

where $V = 3$ is the number of views, and subscripts i and j denote one of three image domains: freehand sketches, line drawings or RGB renderings.

In Fig. 4.1, we plot pairwise distances between shapes, when their views come from one of the three image domains. We can see that in all three configurations, comparing the same object between different domains results in the lowest average distance (darker color) in most of the cases. This shows general robustness of the CLIP model across different domains. We also can see that it is easier to find correct matches for 3D shapes represented with amateur sketches and synthetic sketches than for 3D shapes represented with amateur sketches and RGB renderings.

However, failure cases are possible, in particular, when amateur sketches are concerned: sometimes the distance in CLIP embedding space can not confidently distinguish two shapes in two different domains. Sometimes comparing the same object does not result in the lowest distance or there is overlap between confidence intervals.

Matching views across various domains Considering individual views k and h in domains i and j, we obtain the correspondent view embeddings for the same object and compare them,

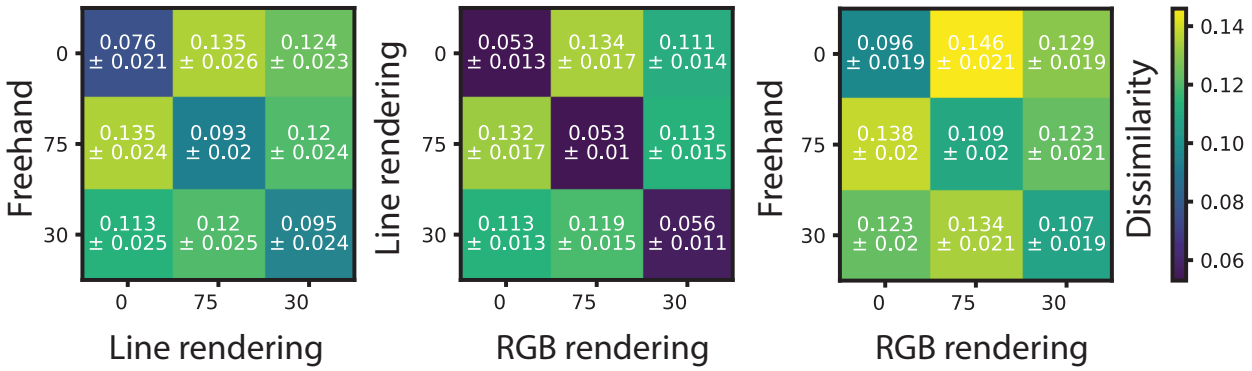


FIGURE 4.2: We plot average pairwise distances between views in different image domains: freehand sketches [5], line or RGB renderings. See Section 4.2.3.1 for the details.

averaging over different objects:

$$d(k^i, h^j) = \frac{1}{N} \sum_{A=\text{obj}_1}^{\text{obj}_N} \left(\text{CLIP}(A_k^i) - \text{CLIP}(A_h^j)\right)^2 \tag{4.5}$$

where $N = 100$ is the number of objects. Fig. 4.2 shows that for all configurations the average lowest values are obtained when comparing the same view. This shows that, in general, CLIP is able to match the same view of an object between different domains. Similarly, we observe that matching views of freehand sketches with line renderings is more reliable than matches of freehand sketches with RGB renderings.

However, when comparing freehand sketch views with views from other domains, we observe that the confidence intervals can overlap. This means, that occasionally an incorrect viewpoint in a different domain can be selected.

Overall, our analysis shows that the distance in CLIP embedding space is a promising perceptual loss, motivating us to further investigate the use of large Visual-Language models for sketch-based 3D tasks. In the following sections, we discuss the strategies that we propose to increase the robustness of matching views of freehand sketches and line renderings.

4.2.3.2 Selecting best setting

For the experiments in this section, unless specified otherwise, we use the set of $0°, 30°, 45°, 75°$ and $90°$ azimuth angles, common for chair and lamp category sketches from the *AmateurSketch* dataset [5], to represent 3D models. We use ResNet-based variant of the CLIP image encoder, unless specified otherwise.

To evaluate retrieval accuracy, we use standard retrieval tasks top-1, top-5 accuracy measures, which compute the percentage of time the ground-truth is returned among top 1 and top 5 ranked retrieval results, respectively.

Sketch and shape projection area In our preliminary experiments, we observed that scaling sketches and 3D model projections to fit the same bounding box area results in improved retrieval accuracy. We provide additional comparisons in Appendix B.3.1.

In this section, we evaluate several scaling factors, and show that it is possible to improve top-1/top-5 retrieval accuracy by roughly 5 points by just carefully selecting the size of the bounding box for an object (Table 4.1). First, we randomly select 50 settings for the size of the bounding box from 53.1% to 98.7% of the image size (224 $\times$ 224 pixels, in our case). Then, we sample uniformly the size across the best setting from the previous experiment: namely we sample 20 settings from 51.6% to 59.2% of the image size. We find setting the object bounding box to be equal to 57.7% from the image results in the best retrieval accuracy, resulting in the bounding box of 129 $\times$ 129 pixels.

#	Method	Chairs		Lamps	
		acc@1	acc@5	acc@1	acc@5
1	Random 50 from: [53.1%, 98.7%]	63.90 $\pm$ 3.33	81.35 $\pm$ 3.08	60.90 $\pm$ 2.82	76.60 $\pm$ 1.70
2	Uniform 20 from: [51.6%, 59.2%]	68.17 $\pm$ 0.77	85.40 $\pm$ 0.69	65.18 $\pm$ 0.57	78.69 $\pm$ 0.61
3	Best: 57.7% (129x129)	**69.17**	**86.00**	**65.67**	**78.83**

TABLE 4.1: Selection of the optimal size of the bounding box for an object in the image. See Section 4.2.3.2 "Sketch and shape projection area" for the details.

Robust features Next, we show that we can increase the robustness of the retrieval model by applying the same affine transformations to all queries and shape views and obtaining a complementary set of embeddings to the one computed from the original queries and shape views. We then average the embeddings for each original view instance and its transformed variant to obtain more robust features.

In our experiments in Table 4.2, we analyze if we can find one set of translation and rotation settings that would maximize the retrieval accuracy. First, we run 20 experiments, where we apply random translations along x- an y-axes (Table 4.2#4). Then, we run another 20 experiments (Table 4.2#5), where the translation vectors are sampled uniformly from the range around the best value identified in the previous setup. We then select the best values according to the retrieval performance on both categories (Table 4.2#6), and each category individually (Table 4.2#7,8). We show that (1) the optimal values across categories are similar and (2) the value optimal for one category results in increased retrieval accuracy for another category (highlighted in gray), compared to random translations (Table 4.2#4).

We perform similar experiments for rotation (Table 4.2#8-12). Finally, we combine the best setting for augmentation with translation and rotation (Table 4.2#13). Such approach allows to increase the top-1/top-5 accuracy by 2-3 points on both considered categories.

We observe that averaging over more augmented versions of a view results in better retrieval accuracy. Therefore, as the best approach, we use the averaging of the feature of the original image, translated image, rotated image and rotated and translated image. We refer to this setting as Augm. (4 imgs.) (Table 4.2#14).

#	Method		Chairs		Lamps	
			acc@1	acc@5	acc@1	acc@5
3	OBB (Opt. bound. box)		69.17	86.00	65.67	78.83
	0.5(CLIP(OBB)+ CLIP(f(OBB))					
4	f	Transl. random (20) $x \in [-10\%224, 10\%224]$, $y \in [-10\%224, 10\%224]$	70.70 $\pm$ 1,27	86.99 $\pm$ 0.98	66.18 $\pm$ 0.85	79.94 $\pm$ 0.86
5	f	Transl. uniform (20) $x \in [21, 6]$, $y \in [-1, -6]$	71.02 $\pm$ 0.80	87.81 $\pm$ 0.33	66.96 $\pm$ 0.72	80.05 $\pm$ 0.24
6	f	Transl. (best both) $(19, -4)$	71.50	88.33	67.50	80.00
7	f	Transl. (best chairs) $(17, -5)$	72.33	88.17	67.00	80.17
8	f	Transl. (best lamps) $(19, -2)$	71.17	87.83	68.16	80.17
8	f	Rot. random (20) $\in [-40°, 40°]$	70.60 $\pm$ 1.47	86.65 $\pm$ 0.95	66.32 $\pm$ 0.99	79.97 $\pm$ 0.72
9	f	Rot. uniform (20) $\in [-30°, -20°]$	70.37 $\pm$ 0.73	87.03 $\pm$ 0.55	66.91 $\pm$ 0.39	80.61 $\pm$ 0.40
10	f	Rot. $-23.5°$	72.33	87.50	67.17	81.17
11	f	Rot. (best chairs) $-23.5°$	72.33	87.50	67.17	81.17
12	f	Rot. (best lamps) $-23.5°$	72.33	87.50	67.17	81.17
13	f	transl. $(19, -4)$ & rot. $-23.5°$	**74.50**	88.33	67.50	81.00
14	Augm. (4 imgs.)		74.00	**89.33**	**69.33**	**82.17**

TABLE 4.2: Robust features computation. OBB denotes that objects in all shape and query views are scaled to fit the bounding box which we found to be optimal in Section 4.2.3.2. For the details of experiments, please refer to Section 4.2.3.2 "Robust features".

Minimum vs. average Table 4.3 shows that taking minimum allows to improve the top-1 retrieval accuracy by approximately 9 and 6 points for chair and lamp categories, respectively. And it improves the top-5 retrieval accuracy by 4 and 10 points for chair and lamp categories, respectively.

	Chair		Lamp	
	acc@1	acc@5	acc@1	acc@5
Average	60.00	82.50	51.50	68.67
Minimum	**69.17**	**86.00**	**65.67**	**78.83**

TABLE 4.3: Comparison of feature selection strategies.

Shape views selection The results from the previous section imply that CLIP can differentiate between viewpoints of the same object, conforming to the study in Section 4.2.3.1. In this section, we study the role of viewpoints selection for the 3D shape representation. We scale objects in all views to fit the optimal bounding box (Section 4.2.3.2 "Sketch and shape projection area"). We first check how adding more viewpoints affects the performance. In particular, we represent 3D shapes with 5 viewpoints with azimuth angles set to $0, 30, 45, 75$ and 90 degrees, or with 7 views from the range between 0 and 90, sample equidistantly. Overall, the performance seems rather stable in this scenario. Top-1 retrieval accuracy for chair and lamp categories improves, while for

the chair category the top-5 decreases and for the lamps slightly increases, as shown in Table 4.4.

We then render 3D shape densely from 7 viewpoints in the interval between -5 and 85 degrees. In this case sketch viewpoints do not match any single viewpoint of a 3D shape. Interestingly, having more viewpoints, even though not matching the ground-truth, still results in increase of top-1 retrieval accuracy for the chair category. This is possibly due to the fact that chair sketch viewpoints are deviating from the ground-truth viewpoint in the used dataset [5]. The rest of the considered accuracy measure decreases for categories but not dramatically.

We then conduct an additional experiment where we add more viewpoints around the most likely sketch viewpoints for a given shape category. Interestingly, for both categories the top-1 retrieval accuracy increases, while the top-5 decreases. We, further investigate this behavior by visualizing retrieval results in Fig. 4.3 for the two scenarios, when the shapes are represented with 3 most-likely views for a given category or with 9 views, including additional viewpoints close to the most-likely viewpoints. We hypothesize that for certain objects with addition of more viewpoints some objects can start to appear more similar in non-corresponding viewpoints, resulting in drop of top 5 retrieval accuracy.

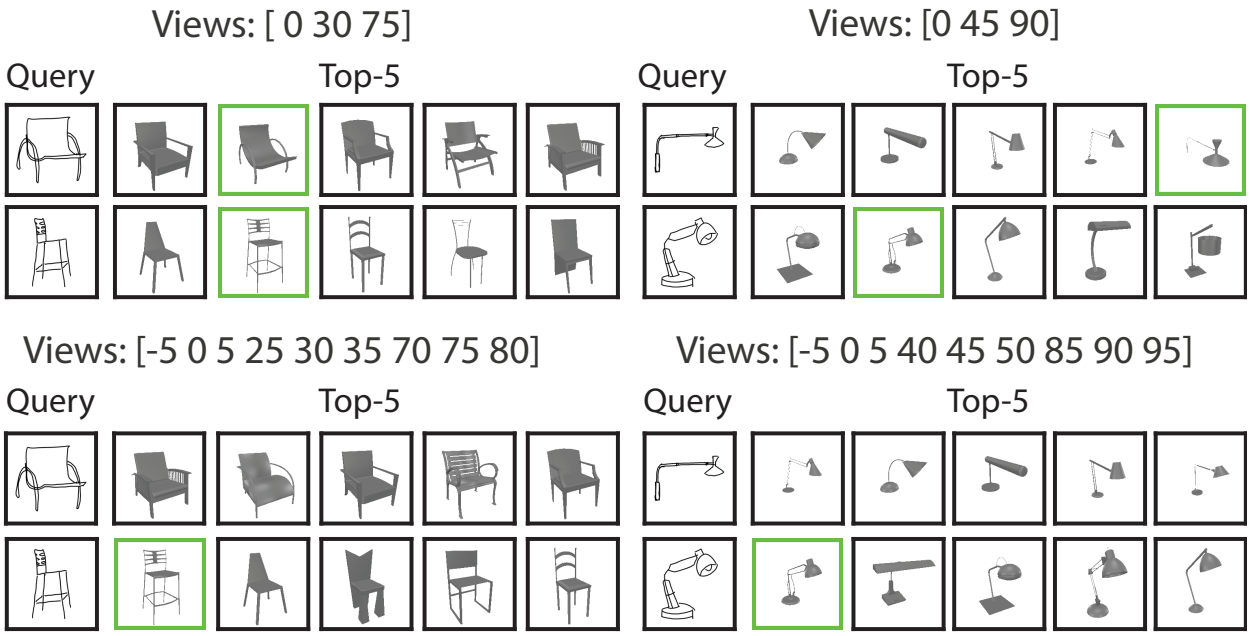


FIGURE 4.3: Comparison of the retrieval results when 3 or 9 views are used to represent 3D shapes. The 1st and 3rd lines show the case when the ground-truth is not among top-5 retrieved results for the 9-view shape representation. The 2nd and 4th lines show examples when the model with the 9-view shape representation can more reliable match the shape, returning it as a top result. The green square highlights the ground-truth 3D shape for each query.

CLIP version & rendering style We compare the ViT- and ResNet-based versions of the CLIP model. We use the sketches from the AmateurSketch dataset [81] as is, and render the synthetic shape views with a fixed distance to the camera, so that the object in all views fits well the image plane. We set the line width to 2.2 in the line renderings. These are the same settings as we used in the experiment in Section 4.2.3.1.

	Chair		Lamp	
	acc@1	acc@5	acc@1	acc@5
Sketch views	68.83	**88.00**	63.33	78.00
$0°, 30°, 45°, 75°, 90°$	69.17	86.00	<u>65.67</u>	**78.83**
7 views: $[0° : 15° : 90°]$	<u>69.83</u>	<u>86.33</u>	**66.00**	<u>78.67</u>
7 views: $[-5° : 15° : 85°]$	69.17	84.33	61.00	74.67
9 views: sketch views $\pm 5°$	**70.16**	83.83	64.83	75.00

TABLE 4.4: Analysis of the retrieval accuracy under different choices of 3D shape views. Boldface denotes the best result in each column, while underline denotes the second best result.

Table 4.5 shows that in our case ResNet-based version gives much better results, while Sangkloy et al. [125] find the ViT-B/16 model to give best results for multi-modal image retrieval from combined sketch and text inputs. In Table 4.5, we also compare the retrieval results when line or RGB rendering is used. It can be seen that line rendering allows to greatly reduce the domain gap.

		Chair		Lamp	
Backbone	Views	acc@1	acc@5	acc@1	acc@5
ViT	rgb;	10.67	21.67	12.17	21.83
ResNet101	rgb;	46.83	66.67	46.17	61.83
ViT	lines;	34.00	56.00	38.83	61.83
ResNet101	lines;	<u>53.83</u>	<u>73.83</u>	<u>51.67</u>	<u>67.83</u>
ResNet101	lines, OBB;	**69.17**	**86.00**	**65.67**	**78.83**

TABLE 4.5: CLIP version & rendering style. Boldface denotes the best result in each column, while underline denotes the second best result. OBB denotes objects scaling in all shape and query views to fit the bounding box we found to be optimal in Section 4.2.3.2 "Sketch and shape projection area".

CLIP and ImageNet pre-training comparison We consider the ResNet101 and ViT architectures and compare CLIP pre-trained models with ImageNet pre-trained models on the test set of the AmateurSketch dataset [81]. In this experiment we center and scale views to fit the bounding box of size 170×170. This corresponds to the smallest square bounding box that fully encompasses objects in the original sketches. We do not use settings studied on CLIP, as OBB, for a fair comparison. In table 4.6 we note that CLIP pre-training achieves better results for ViT, while ImageNet pre-training seems a bit better on ResNet. While CLIP pre-training does not seem to represent a performance advantage on all the architectures in this comparison, having the possibility to naturally extend this approach to hybrid sketch-text queries in the future, due to the Vision-Language pre-training of CLIP, represents a strong reason to prefer this model. In fact, sketches and text can be combined to convey complementary information. Thus, we decide to use the CLIP image encoder.

	Chair		Lamp	
Backbone	acc@1	acc@5	acc@1	acc@5
ViT CLIP	30.51	52.57	48.05	69.07
ViT ImageNet	27.70	50.08	38.44	57.06
ResNet101 CLIP	60.03	79.10	66.37	80.78
ResNet101 ImageNet	63.02	78.60	66.96	82.58

TABLE 4.6: CLIP and ImageNet pre-training comparison on the test set of the AmateurSketch dataset [81].

4.2.3.3 Learning CLIP features masking or weighting

In this experiment, we use a training dataset of 4475 ShapeNet chair models. We generate line renderings as we did for the rest of the experiments. We render five different views for every example: 0, 30, 45, 75, and 90, obtaining 4475×5 line renderings. During training to increase generalization to freehand sketches, we augment 3D shape views with random affine transformations. We provide implementation details in Appendix B.5.

During training we monitor the performance on the validation set of freehand sketches. While the training loss on the synthetic sketches consistently decreases, the performance on freehand sketches first goes down and then goes up. Therefore we use a small set of freehand sketches to use the best checkpoint, and then show that this generalizes across the two considered shape classes and to the test sets of sketches. In particular, we select the best checkpoint on one class and then test on another class, and vice versa.

	chairs (200)		lamps (200)	
	acc@1	acc@5	acc@1	acc@5
Augm. (4 imgs.)	74.00	89.33	69.33	82.17
Binary (best chairs)	**77.67**	**91.83**	70.00	83.17
Binary (best lamps)	76.16	91.50	**70.83**	**83.33**
Continuous (best chairs)	77.50	91.67	69.67	83.17
Continuous (best lamps)	75.33	91.67	70.50	83.17

TABLE 4.7: Comparison of our learning-based feature masking and weighting strategies Section 4.2.2.2. The first line represent the baseline without feature selection. The gray highlights the test class. Boldface highlights the best result.

Masking vs. weighting Table 4.7 shows that the difference in performance between masking and weighting is not strong. However, we find that masking results in slightly better performance and generalization to a different shape class.

4.2.3.4 Comparison with the previous supervised work on the test set

Table 4.8 shows that already the baseline CLIP-based retrieval model results in similar performance to the supervised approach by Qi et al. [5]. We, further, show, that our careful choice of settings and feature masking strategy result in the new state-of-the-art retrieval accuracy. In particular, we improve the top-1 retrieval accuracy by roughly 18 and 17 points for chair and lamp categories, respectively, compared to the previous state-of-the-art. For the method by Qi et al. [5]

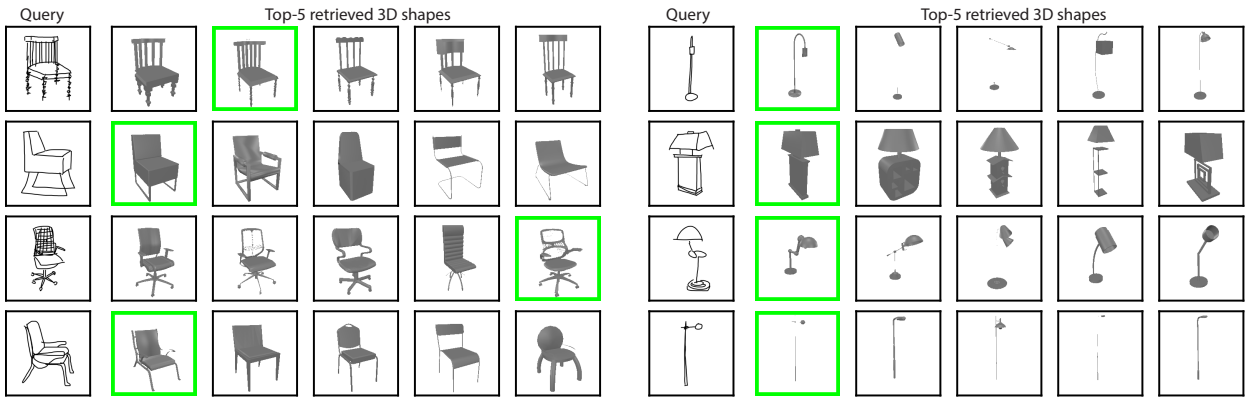


FIGURE 4.4: Random samples of the retrieval results of our method, where we use scaling and centering to fit the optimal bounding box, the robust feature computation from original views and their 3 additional augmentations, and the learned binary masking of non-relevant features. The green square highlights the ground-truth 3D shape for each query. Note that the sketches in this figure are not scaled for improved visualization.

the numbers are taken from their paper. Table 4.8 also shows strong generalization properties of our approach. Note that our learning-based method uses only synthetic sketches (line rendering) of chair class, and even if we select the checkpoint based on performance on freehand sketches from the chair class, the performance on the lamp class still exceeds the one of the previous supervised method.

Fig. 4.4 shows visual quality of the retrieval results of our method. We show random examples from the test set. We can see that most of the times our approach is able to select the correct shape as first choice or between the first five choices, consistently with the quantitative results. We also note that between the first five choices, many shapes are similar to the query sketch and resemble his features.

	chairs (201)		lamps (111)	
	acc@1	acc@5	acc@1	acc@5
Qi et al. [5]	56.72	87.06	57.66	87.39
Baseline	51.58	70.81	54.65	73.57
OBB	65.17	82.09	70.27	82.58
OBB, Augm. (2 imgs)	68.65	84.07	72.37	84.68
OBB, Augm. (4 imgs)	70.64	85.24	74.17	86.49
Binary (best chairs)	**75.95**	**88.39**	74.17	**88.59**
Binary (best lamps)	75.12	87.89	**74.74**	87.69

TABLE 4.8: Comparison of our work with the previous supervised approach for instance-level sketch-based 3D shape retrieval on the test set of sketches from [5]. *OBB*, represents the setting where objects in all views are centered and scaled to fit 129×129 pixels bounding box. *OBB, Augm. (2 imgs)* stands for the baseline where the images are first converted to fit the optimal bounding box, and then the features from the original and transformed images are averaged (Section 4.2.3.2). And in *OBB, Augm. (4 imgs)*, we apply three different augmentations (Section 4.2.3.2). Finally, we combine *OBB, Augm. (4 imgs)* with the feature masking approach, where the checkpoints are selected on the validation sets of the specified in braces classes (Section 4.2.3.3). Boldface denotes the best results, and the second best results are underlined.

4.2.4 Conclusion

We investigated the effectiveness of a large pre-trained model, CLIP, as a perceptual loss when applied to discriminate objects and views between different domains showing promising results. We then studied the usage of CLIP for sketch-based shape retrieval. We proposed a methodology to study the role of different factors to the retrieval task, and find a set of optimal settings. We also introduced a learning-based strategy to mask certain CLIP features that might be non-relevant to a particular downstream task: sketch and line rendering comparisons, in our case. Such strategy does not modify the original model weights. It can be also found beneficial for multi-modal retrieval, where one might want to use spatial sketch information in isolation of color attributes, provided by text inputs, which we leave for future work. With this work, we established a new state-of-the-art in instance sketch-based 3D shape retrieval. Compared to the previous work, we need just a very small set of paired freehand sketches and 3D shapes for one shape class. We show cross-generalizability of the best settings across two shape classes for which the instance-level pairing between 3D shapes and 2D sketches is available.

Overall, we succeed in applying pre-trained Vision-Language models to sketch-based 3D shape retrieval without using supervised data for training, showing improved performance compared to the previous state-of-the-art supervised work.

4.3 CLIP-based 3D recontruction from sketches

In the following, the investigation on Vision-Language models applied to sketch based 3D tasks continues analyzing 3D reconstruction from sketches. A CLIP perceptual loss is applied to an optimization-based solution for sketch based 3D reconstruction to study alternatives to the use of supervised datasets, a strong obstacle for 3D tasks based on freehand sketches. The investifation starts with the definition of the framework. An approach based on the inversion of a pre-trained 3D StyleGAN is proposed. This solution fits well the need of an unsupervised method, it makes possible to define a CLIP perceptual loss between the rendering of a generated 3D shape and a freehand sketch. We report experiments, considering inversion with 3D supervision, validating the framework, and freehand sketch supervision, where generated shapes show characteristics that resemble the target sketches. Finally, a conclusive discussion is presented about the obtained results and how it is possible to expand the investigation with future extensions, i.e. adding regularization to get more realistic shapes, specializing the CLIP loss for sketches and adding constraints to improve the optimization procedure.

4.3.1 Related Works

The StyleGAN architecture [94, 95, 96] and the StyleGAN inversion technique [103, 104, 133, 134, 135] are relevant components of the proposed solution. In the StyleGAN paper [94], the authors propose a new GAN encoder, leaving all the other GAN settings unchanged (discriminator, loss, regularization, etc.). The new encoder predicts different style vectors that affect the encoder generation at different levels, controlling the features at different scales and leading to unsupervised separation of high level attributes and stochastic variations. The input latent code, that previously

was directly given as input to the encoder, now is embedded in an intermediate level that is less bounded to the probability density distribution of the training data, favoring a more disentangled representation of different factors of variation. Overall, the new architecture sets a new level of image generation quality and shows a better control over factors of variation. The StyleGAN architecture was further improved eliminating and reducing known problems of the first version [95, 96].

The GAN inversion technique regards the embedding of a given example, often an image, to the latent space of the generative network. After the embedding of an image, it is possible to apply editing operations in the latent space and then generate a new image from the modified latent code. In [103] the inversion of the StyleGAN architecture is studied considering different embedding spaces and showing editing results on morphing, style transfer and expression transfer. Many other works investigated further this line of research [104, 133, 134, 135].

In our project, we define an inversion framework based on a pre-trained 3D StyleGAN architecture. We aim to embed sketches in the latent space of the generative network with a loss defined on the embedding space of CLIP. Then, we can generate the 3D shape corresponding to the obtained latent code, investigating 3D shape reconstruction from sketches in an unsupervised manner.

Many previous works proposed solutions for 3D shape reconstruction from sketches [87, 88, 89, 11, 90, 10, 12, 7, 91, 8], in Section 2 we discuss some of the solutions that have been proposed. Some of the previous methods trains on {synthetic sketches, 3D shape} pairs [89][11][12][7], that do not require datasets with 3D shapes labeled with freehand sketches, but suffer from a domain gap when applied to freehand sketches. The work from Kong et al. [9] achieves good reconstructions from freehand sketches training a diffusion model on a supervised dataset. However, the obtained results are limited to the lamps and chairs categories, that correspond to the classes of the largest available supervied dataset, strongly limiting the applicability of the proposed solution. A recent solution [92] recovers 3D missing information from sketches with an image translation network before predicting a density map, then a 3D point cloud is generated sampling (x, y) coordinates from the density map and z coordinates from the obtained 2D points. This approach is able to obtain good reconstructions considering sketches with reliable shapes, while deformations in the sketch also appears in the generated 3D shape.

We, instead, analyze the use of a large Vision-Language model, CLIP, to overcome these limitations, studying if the ability of CLIP to generalize between domains can be applied to reconstruct 3D shapes from freehand sketches without the use of a supervised dataset.

4.3.2 Method

4.3.2.1 3D StyleGAN Inversion

We consider an unsupervised sketch based 3D reconstruction solution based on the inversion of a pre-trained 3D StyleGAN generator. A 3D styleGAN generator [136], similarly to the 2D counterpart [95], is composed of N subsequent style layers, where every layer receives as input the output of the previous layer, a style vector and random noise. The style vectors are obtained

from different affine transforms applied to the feature vector W, where W is obtained applying an MLP to the random gaussian noise z.

We decide to rely on the recent introduced SDFs as 3D representation due to their advantages over previous solutions. In particular, in this implementation, the 3D generator predicts a feature volume first. Then, given a 3D coordinate $x \in [-1, 1]^3$, a trilinear interpolation of the feature volume is achieved to obtain the feature vector $\Phi(x)$. To obtain the SDF value for x, a simple MLP, takes as input $\Phi(x)$ and predicts a signed distance value, $SDF(x) : mlp(\Phi(x))$, where $x \in R^3$ and $SDF(x) \in R$. For further details on the 3D StyleGAN see [136].

The inversion of a generative network consists in finding the latent/random vector or input in general that makes the fixed generative network to output a desired target example. Implementation choices include the representation of the target example (e.g. 2D or 3D), the level at which the inversion takes place, i.e. the latent vector we want to find a value for (random noise z, feature space W or many others), the type of inversion and the loss used for the inversion procedure.

We choose to invert at the level of the style vectors, the ones that are given as input to the style layers. We start inverting using an SDF example as supervision and then we move to sketch based inversion. When a sketch is considered as target example, an SDF differentiable renderer [137] is used to get renderings from the generated SDF, so as to compare the target sketch with a 2D representation of the generated shape. The defined loss vary depending on the nature of the target example, a 3D or a 2D loss is defined depending on the supervision. Optimization, learning and hybrid are all possible inversion approaches, in this investigation we will consider the optimization one, that does not require any type of training procedure.

4.3.2.2 Optimization based inversion

The optimization inversion initializes a starting latent vector, in our case the style vectors, and defines an optimization based procedure to iteratively update it so that the generated SDF resembles a target 3D shape. In our project we consider two different representations of the target 3D shape: SDF and freehand sketches. Any target 3D representation, like the SDF in our case, allows to guide the inversion directly comparing the generated 3D shape with the target one, representing a very strong supervision. Before diving in the sketch based inversion, we start from this simpler setting to verify that the inversion pipeline works properly.

Given a target 3D shape represented as an SDF, SDF_{gt}, the style vectors s and the pre-trained 3D StyleGAN $SGAN$, the loss for the optimization procedure is defined as follows:

$$L\_optim_{3D}(s) = MSE(SDF_{gt}, SGAN(s)) \tag{4.6}$$

where $SGAN(s)$ is the generated SDF and MSE corresponds to the mean squared error metric. This loss is used to update s to generate an SDF that is more similar to SDF_{gt}. The process requires multiple iterations to converge.

The freehand sketch of a taget 3D shape correspond to the 2D representation we are interested in, an amateur freehand sketch. Inverting a 3D GAN with this type of sketch makes possible to find 3D shapes that resemble simple sketches that any non-professional could draw. It is not possible to directly compare a 2D sketch with a 3D shape. As previously anticipated, we will

use an SDF renderer to represent the generated 3D shape as a 2D rendering. We now have two images, the sketch and the rendering, that can be compared with a simple regression loss like MSE. However, using a loss like MSE, will not work well on these images, sketches and renderings present a domain gap that compromise the effectiveness of a simple per-pixel comparison. The rendering of a 3D shape is a precise 2D representation of the object, where the shape and the texture are faithful to the starting 3D shape. An amateur sketch, instead, is sparse, without texture or colours, abstract, the object may not be represented in a realistic way, and his lines are hand drawn, so they may not be precise. We need a loss that takes into consideration the semantic and the general/high level geometry of the object depicted in an image, independently from the type of the image itself (like natural image, rendering or sketch). We need a powerful perceptual loss. Motivated by the promising results obtained in the previous retrieval Section 4.2, we further investigate the use of CLIP as a perceptual loss to compare the amateur sketch $sketch_{ama}$ depicting the target 3D shape with the rendering of the generated 3D shape:

$$L\_optim_{ama}(s) = MSE(CLIP(sketch_{ama}), CLIP(Rend(SGAN(s), rend\_p))) \tag{4.7}$$

where CLIP indicates the features obtained by the third layer of the pre-trained ResNet101 visual encoder of CLIP [13], *Rend* is an SDF differentiable renderer and *rend_p* are all the information needed to set up the rendering operation, like the distance of the camera, azimuth and elevation.

Independently from the target 3D shape representation, the optimization based inversion requires an initialization method to define the starting value for the style s. The pre-trained 3D StyleGAN is able to generate random shapes from the learned distribution. Sampling 1000 shapes, saving the s values for every shape and obtaining the average is the method we decided to use to initialize s. The use of this approach is widespread in the literature and it makes sure that the starting point is in the portion of the space that returns reasonable results, so that navigating the space through optimization is less likely to end up in a point that returns bad generations.

4.3.3 Experiments

In this section we experiment with the introduced inversion approach to investigate the use of Vision-Language models, in our case CLIP, to define a perceptual loss for 3D shape reconstruction from freehand sketches in an unsupervised manner. In the following we will first talk about the dataset we use and then introduce our experiments with the optimization technique starting from the 3D SDF supervision and then moving to the freehand sketch based inversion.

4.3.3.1 Dataset

In our experiments we make use of an existing dataset of 3D meshes paired with amateur freehand sketches, the *AmateurSketch* dataset [5]. This is the same dataset used for the retrieval work, where sketches are drawn by participant without previous sketching experience. The elevation angle for the drawing is fixed to 20 degrees and three values of azimuth are defined for each category (0°, 30° and 75° for chairs, 0°, 45° and 90° for lamps). We enrich the dataset with SDFs obtained from the 3D meshes, in particular they are the same used in the 3D StyleGAN work [136]. We consider

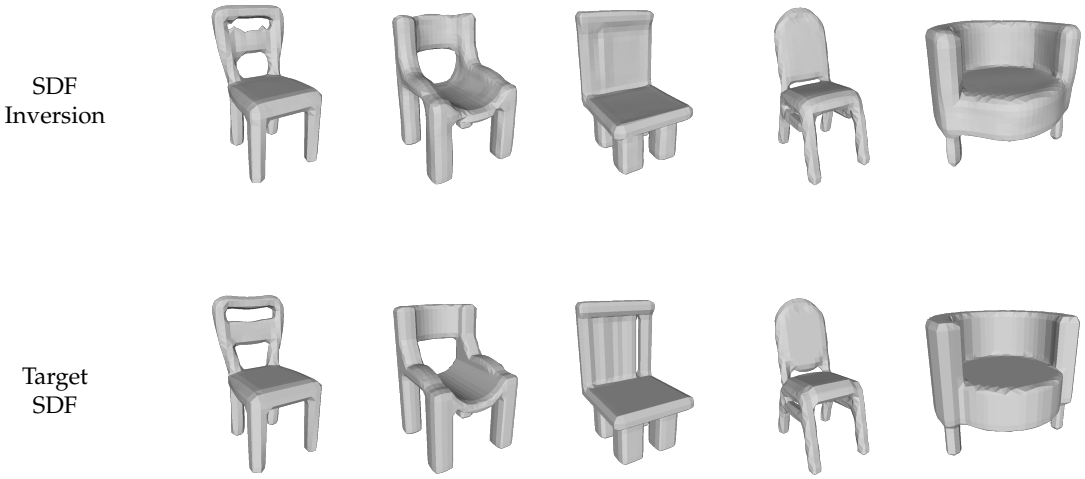


FIGURE 4.5: Optimization based inversion results using a target SDF to obtain a similar 3D shape from a pre-trained 3D StyleGAN. Target and obtained SDFs are both shown as meshes.

the examples from the chair category from [5], where the 3D meshes are in turn obtained from the ShapeNetCore dataset [130].

4.3.3.2 Implementation Details

The experiments are based on the optimization based inversion described in Section 4.3.2.2 where the details about data and parameters are provided in the following. The optimization is run 10 times for every experiment, every time with a different example. We fix 10 random examples from the chair category of the presented dataset in pairs of {freehand sketch, SDF}. We select the 0° azimuth sketches (front view) from the dataset and use elevation 20° and azimuth 0° for the renderings. The batch size is set to 1. The learning rate is set to 0.005 and the optimizer is Adam. When using SDF supervision the number of iterations is set to 5000, the more the number of iterations the better is the result. The experiments with CLIP loss and 2D supervision are run for 1000 iterations, we notice that, for most of the examples, after that amount of iterations there are no further improvements.

4.3.3.3 Optimization inversion with 3D supervision

In the first experiment we initialize the latent vectors with the average initialization previously introduced and compare the generated SDF with the SDF_{gt} to define the optimization loss, as shown in equation 4.6. We select the best generated shape to be the one with the lowest loss value.

Results are shown in Fig. 4.5. In all the figures SDFs are transformed in meshes through the marching cubes algorithm [138] for visualization purposes. The top row shows the shapes obtained with inversion, while the bottom row shows the target 3D shapes. Using SDFs as supervision the inversion is able to return 3D shapes that resemble the target ones in most of the cases. This result is promising, it shows how the 3D StyleGAN latent space is populated with

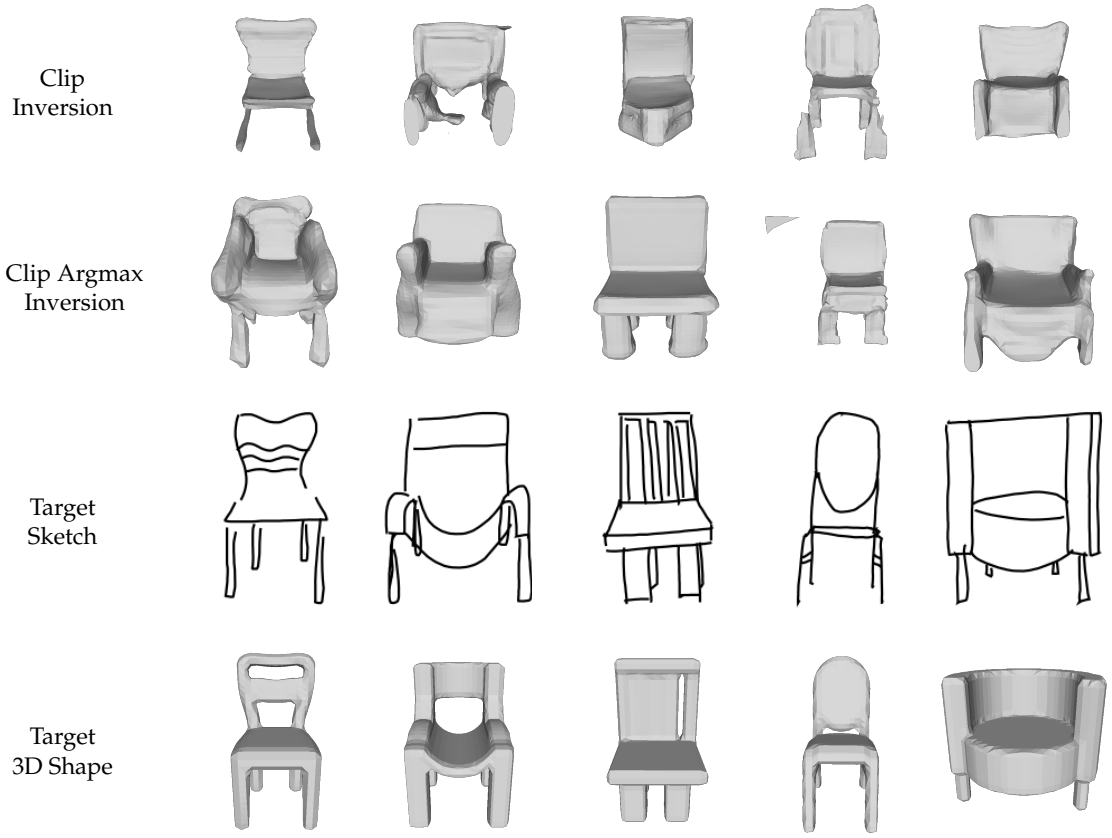


FIGURE 4.6: Inversion results as front renderings of 3D meshes. Lowest Clip loss shapes are picked. From top to bottom row we show Clip Inversion meshes, Clip Inversion meshes where the embedding dimension is reduced with the argmax operation, the target sketches used for the inversion and the target shapes that were used to draw the target sketches.

different types of chairs and how it is possible to navigate such space to find a chair of interest given a strong supervision. The inversion approach works well and represents a good setting to investigate the use of CLIP as a perceptual loss for sketch based 3D reconstruction.

4.3.3.4 Optimization inversion with 2D supervision

In this experiment we initialize the style vectors s with the average technique and we invert the pre-trained 3D StyleGAN using freehand sketches. We compare the rendering of the generated SDF $Rend(SGAN(s), rend\_p)$ with the target amateur sketch $sketch_{ama}$ as introduced in equation 4.7 of Section 4.3.2.2. While optimizing, we select as the best generated SDF the one with the lowest loss value.

Fig. 4.6 and 4.7 show the obtained results from the front and lateral view respectively, the shapes are the same in these figures, they are just rendered from different views. In the figures, the first (Clip Inversion), third (Target Sketches) and fourth (Target 3D Shapes) rows show respectively the renderings of the obtained 3D shapes, the target sketches and the renderings of the 3D shapes that are paired with the target sketches. It is useful to remember that the optimization is guided from a single sketch that correspond to the front view, so no information about how the shape should look from other views is given.

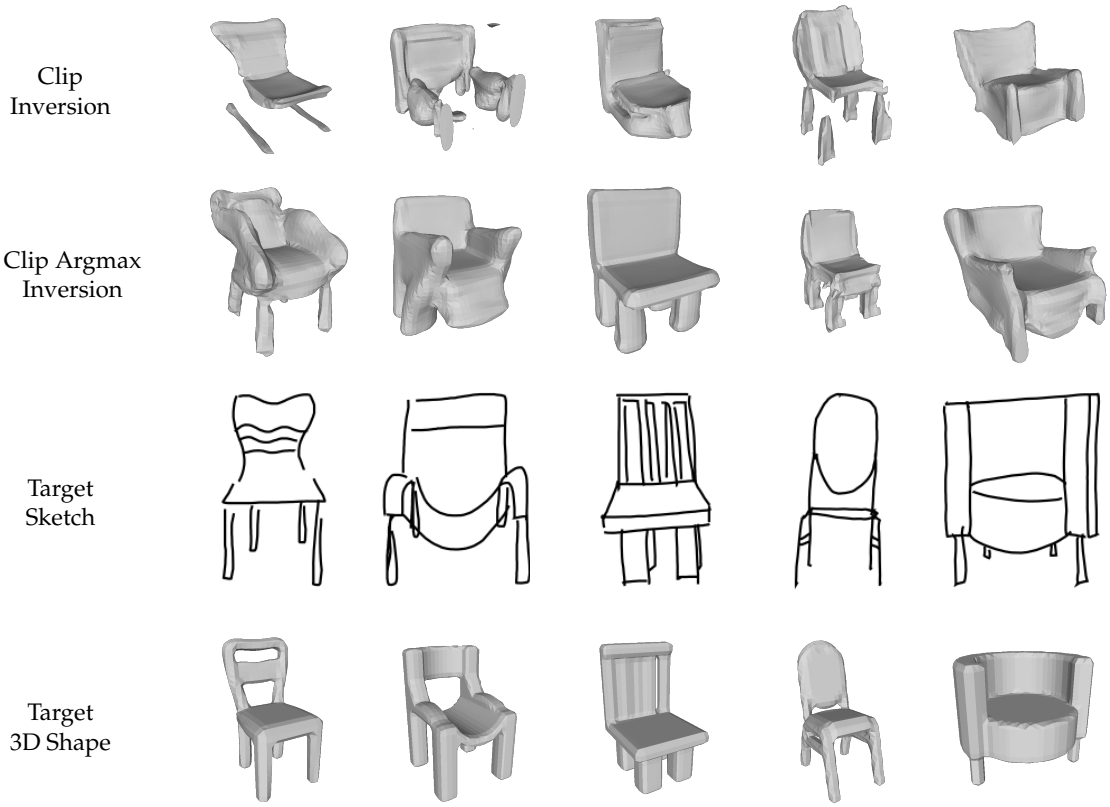


FIGURE 4.7: Inversion results as lateral renderings of 3D meshes. Lowest Clip loss shapes are picked. From top to bottom row we show Clip Inversion meshes, Clip Inversion meshes where the embedding dimension is reduced with the argmax operation, the target sketches used for the inversion and the target shapes that were used to draw the target sketches.

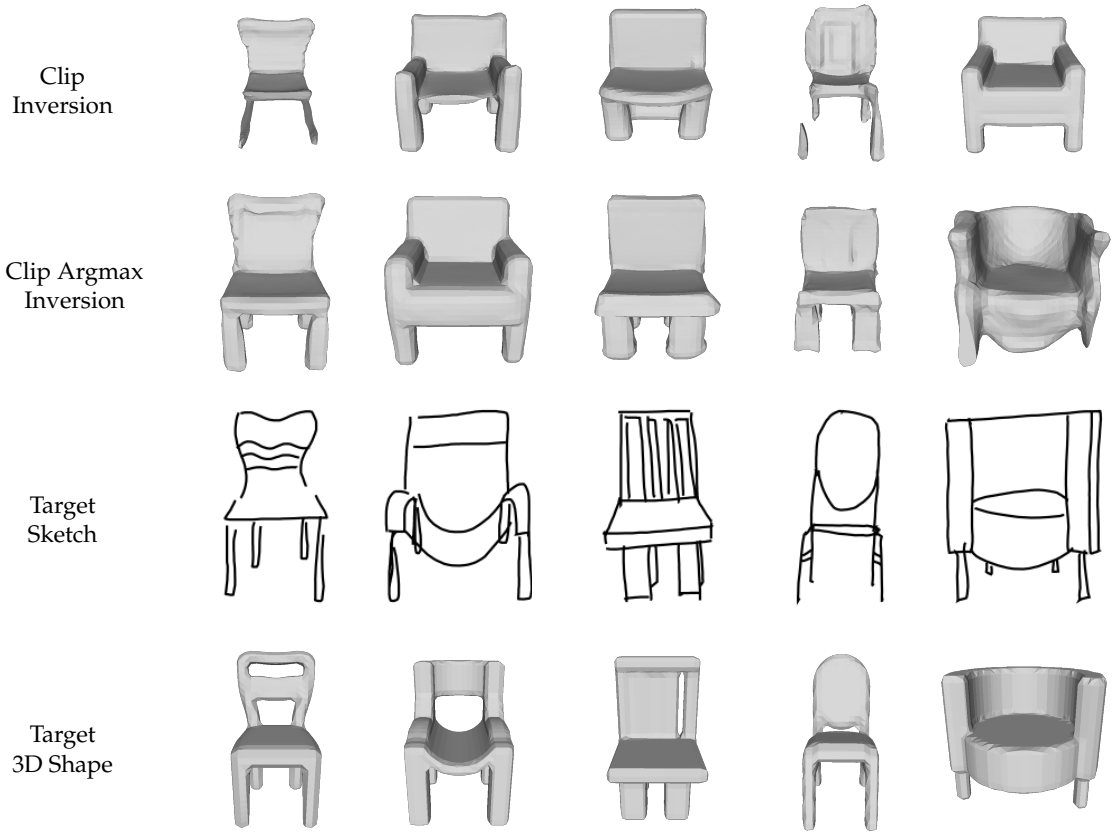


FIGURE 4.8: Inversion results as front renderings of 3D meshes. Shapes that best represent sketches are human picked during optimization. From top to bottom row we show Clip Inversion meshes, Clip Inversion meshes where the embedding dimension is reduced with the argmax operation, the target sketches used for the inversion and the target shapes that were used to draw the sketches.

The results show that there are some problems in the generated shapes, for example missing or bad reconstructed parts, but they also show how many generated shapes resemble characteristics that are present in the target sketch. In the first column from the left, the back and the sit of the generated chair have a shape that are similar to the sketch ones. In the fourth column, the back in the generated shape presents a round characteristic, even if the shape is not the same as in the sketch, the round feature is preserved. Also in column 2 and 3 we note that the general shape of the back resemble the sketch one.

Looking at the results at different iterations, i.e. generated shapes that does not have the lowest loss value, we find that better shapes are generated during optimization. In Fig. 4.8 and 4.9 are shown human selected generated shapes for the same experiment. We save shapes every 25 iterations and we select the best one considering both fidelity to the sketch and realism of the generated shape using our human judgment. We see how, comparing with lowest loss selection, we are able to find more realistic shapes in general, see columns 2 and 3.

The lowest loss value is not always able to identify between the generated shapes the one that best resembles the target sketch. One factor that could contribute to this behaviour is the information represented in the CLIP embedding. CLIP is trained on many natural images, where background, light, texture and colours are relevant. These information are useful in general but

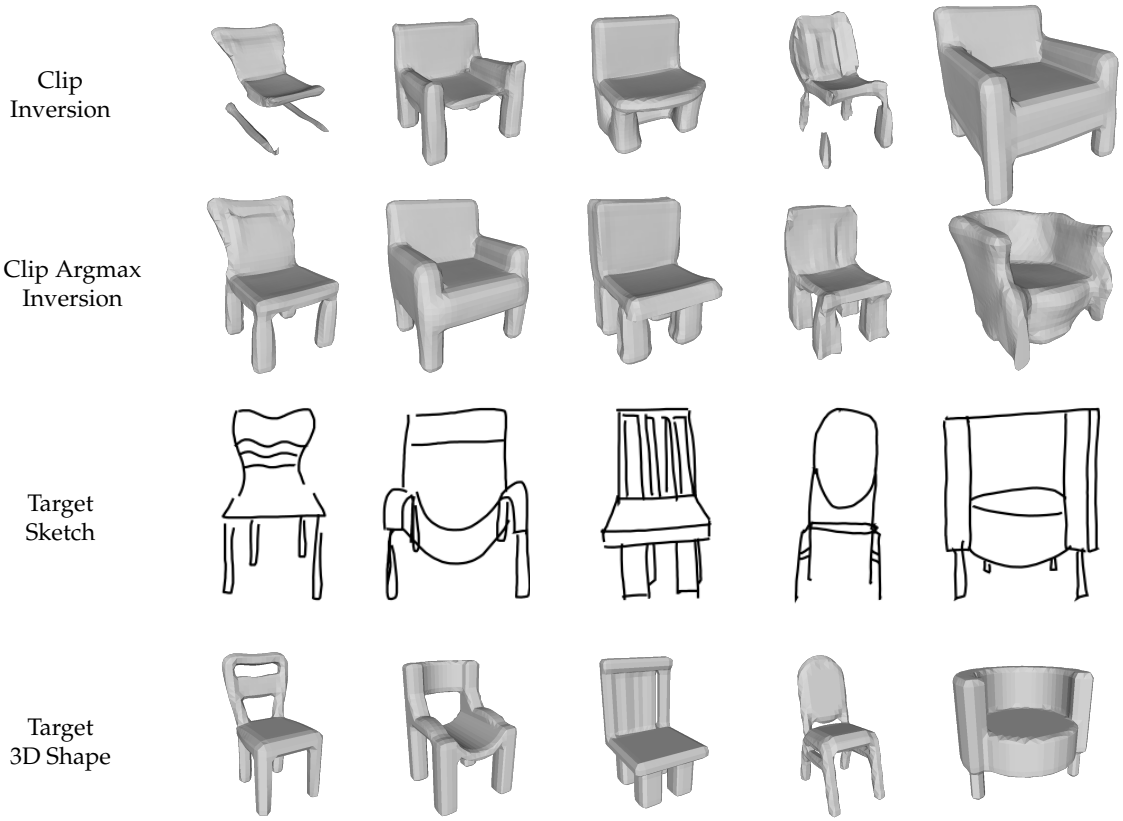


FIGURE 4.9: Inversion results as lateral renderings of 3D meshes. Shapes that best represent sketches are human picked during optimization. From top to bottom row we show Clip Inversion meshes, Clip Inversion meshes where the embedding dimension is reduced with the argmax operation, the target sketches used for the inversion and the target shapes that were used to draw the sketches.

not when dealing with sketches. Motivated also from the results obtained in the retrieval work while learning to mask or weight CLIP features, we replicate this inversion experiment reducing the dimensionality of the CLIP embedding in a very simple way. We apply an argmax operation to the CLIP embedding of both, the rendering of the generated shape and the target sketch, reducing the dimensionality of the embedding by a factor of 4 before comparing them. Following our intuition, we want to investigate if a simple simplification of the embedding space is already able to show improvements when an optimization procedure is concerned. Note that this is a different setting compared to the retrieval experiment, here the comparison between images based on the CLIP embedding defines a loss that is used for optimization. We did not directly apply the previously masked/weighted weights because they are specifically obtained for retrieval, where the weighted embeddings are not used for optimization. We leave the definition of a procedure to mask/weight CLIP embeddings when used to define a loss to future work.

In row 2 (Clip Argmax Inversion) of Fig. 4.6, 4.7, 4.8 and 4.9 are shown the results of the replicated experiment. In Fig. 4.6 and 4.7 the best shapes are chosen as the ones with lowest loss value, while in Fig. 4.8 and 4.9 the best shapes are human picked in the same way as before.

Comparing the lowest loss value shapes (Fig. 4.6 and 4.7) between the experiment without argmax (row 1, Clip Inversion) and the argmax experiment (row 2, Clip Argmax Inversion), we

note that adding the argmax the obtained shapes miss less parts and seem more realistic in general, while still keeping some of the features that resemble the target sketches. The same comparison on human picked shapes show more realistic shapes in columns 1 and 4, while also showing shapes that better resemble the target sketch in column 1, 3 and 5.

4.3.4 Discussion and future work

In this work we investigate the use of Vision-Language models, in particular CLIP, as a perceptual loss for 3D reconstruction from amateur freehand sketches without the need of training with supervised data. We define an inversion framework and show experiments with SDF and freehand sketch supervision. Results show how using optimized based inversion of a pre-trained 3D StyleGAN with a 3D supervision, in our case SDFs, it is possible to obtain shapes that resemble very well the target shape, validating the proposed framework. When changing the supervision with a CLIP loss comparing the target freehand sketch and the rendering of the generated SDF, obtained shapes are not precise and may have defects but they also show characteristics that resemble the target sketches. It is important to note the simplicity of the defined CLIP loss, where CLIP is used as it is and no additional contribution to the loss is added. Being already possible to transfer some features from the target freehand sketch to the inverted 3D shape is a promising result to further pursue on this research direction.

This investigation can be extended in many ways. The realism of the obtained shapes can be enhanced adding a regularization term to the loss. It is possible to add the L1 of the distance between the average initialized style vectors and the current vectors, this could help to keep the style vectors in a region of the space with reasonable shapes. An alternative is to add a contribution to the loss based on a discriminator of SDFs, to keep the generated shapes realistic. The results from the argmax reduced CLIP embedding show results that motivate further investigation. Working on selecting the most meaningful features when using CLIP as a perceptual loss working on sketches could be another way to improve the optimization while it could also effect other sketch research fields where CLIP losses are applied. Finally, additional constraints to the CLIP loss could help to limit the search space while optimizing, concentrating more on the right portion of the space. While our experiments use only one point of view for the CLIP loss, it is possible to experiment with multiple sketches from different points of view for the same example. To improve the CLIP loss, it is also possible to augment the target sketch and or the rendering of the generated shape to define multiple comparisons even with a single view or combine CLIP losses based on different image encoders, like ViT and ResNet101, or multiple layers of the same CLIP image encoder, as done in [132].

4.4 Conclusion

In this chapter we investigated and successfully applied Vision-Language models to sketch-based 3D tasks to overcome the domain gap between freehand sketches and other image domains, without relying on supervised datasets for training. In particular, we conducted a study on the use of CLIP as a perceptual loss, showing promising results for objects and object views discrimination between different domains. We presented a state-of-the-art solution for sketch-based 3D shape

retrieval based on CLIP, a large pre-trained model. In this solution we propose a methodology to find optimal settings for retrieval and a learning strategy to consider only relevant CLIP features to a downstream task. We concluded extending the investigation to the 3D reconstruction from sketches task. We considered an inversion approach based on a pre-trained 3D StyleGAN proposing a CLIP perceptual loss between sketches and renderings as inversion guidance. We exhibit promising results where obtained 3D shapes showed characteristics that resemble the target sketches and discussed multiple future extensions: regularization terms for realism and reducing CLIP loss embedding space or adding constraints to improve optimization.

Chapter 5

Learning the Space of Deep Models

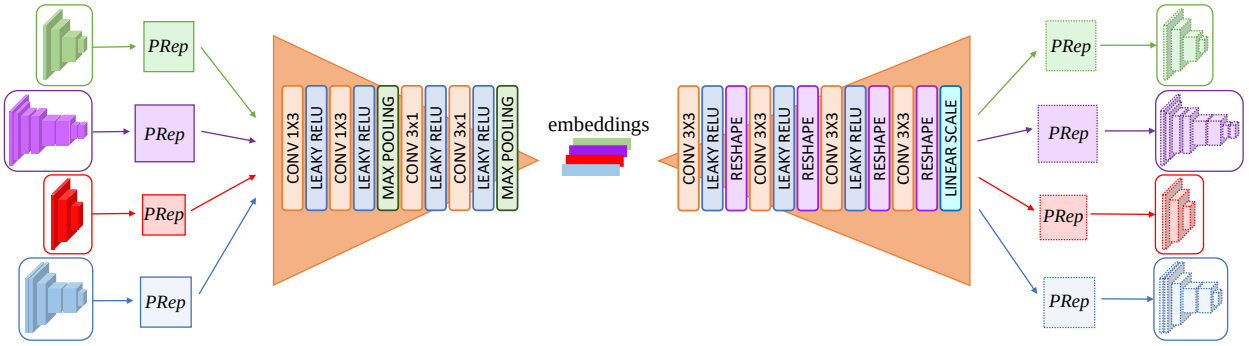


FIGURE 5.1: Overview of the NetSpace framework.

The majority of the time and resources of the PhD were dedicated to the main topic of the thesis: sketches and geometric information, treated in the previous chapters. All the remaining effort was concentrated on a parallel project. A different research direction was pursued in collaboration with a colleague with similar experience.

The last chapter is dedicated to this project, where a more general machine learning topic is investigated. In a few words, this work is about learning the latent space of neural networks and navigate such a space to search for ready-to-use networks. Despite the most promising practical applications have still to be identified, we think that this work holds the potential to have an impact on the future of the deep learning research. This project represents just the first step in this research direction, nevertheless we are very proud of the final result and we hope it will result in an enjoyable reading.

5.1 Project Introduction

Representation learning has achieved remarkable results in embedding text, sound and images into low dimensional spaces, so as to map semantically close data into points close one to another into the learnt space. In recent years, deep learning has emerged as the most effective machinery to pursue representation learning, many scholars agreeing on representation learning laying at the very core of the deep learning paradigm. On the other hand, the success of network compression and pruning approaches [139] highlight the redundancy of parameters learned by a deep learning model, as in the Lottery Ticket Hypothesis [14], which shows that training as few parameters as 4% of those of the full network (i.e. the *winning tickets*) can attain similar or even higher performance.

Thus, we felt puzzling and worth investigating whether the parameter values of a trained deep model might be squeezed into a semantically meaningful low-dimensional latent space. Two questions arise: is it possible to train a deep learning model to learn to represent other, already trained, deep learning models? And according to which trait should two already trained models lay either close or further away in the latent space? The Lottery Ticket Hypothesis may suggest the existence of a low-dimensional key set of information that is shared by all possible sets of parameters for a predefined architecture that achieve comparable performance on a given task. Hence, it seems reasonable to conjecture that one might pursue learning of an embedding space shaped according to similarity in performance. Moreover, many recent works have demonstrated how small deep networks can be trained to fit accurately complex signals such as images[140], implicit representations of 3D surfaces [141, 142] and even radiance fields [143]. One might then be willing to embed such models into a space amenable to capture the similarity between the underlying signals.

In this project, a first investigation along this new line of research is proposed. In particular, we show that it is possible to deploy a basic encoder-decoder architecture to learn a low-dimensional latent space of deep models and that such a space can be shaped so to exhibit a semantically meaningful structure. We posit that the loss to drive the learning process of our encoder-decoder architecture should entail functional similarity - rather than proximity of parameter values - between the input and output models. Accordingly, we train our architecture by knowledge distillation to drive the output model generated by the decoder to mimic the behaviour of the input model. In this study we address two settings: learning a latent space from a training set of models with the same network architecture and different parameter values as well as based on a training set comprising models with different architectures. In both settings, we show that the learnt latent space does posses a semantic structure as it is possible to sample new trained models with predictable behaviour by simple interpolation operations. Moreover, we show that in the Multi-Architecture setting a latent space trained on a set of architectures can generate already-trained models of architectures never seen instantiated at training time. Finally, we show that in both settings it is possible to train an architecture by performing latent space optimization on the low dimensional embedding space instead of optimizing directly the full set of parameters.

5.2 Related Work

Representations. Representation learning concerns the ability of a machine learning algorithm to transform the information contained in raw data in more accessible form. A common algorithm is the autoencoder [144], a self-supervised solution where the representation is learnt by constraining the output to reconstruct the input. Our architecture is inspired by the autoencoder but aim at producing outputs that behave akin to the input (e.g. similar performance on a certain task). In a meta-learning paper, LEO [145], the embedding of the weights of a single layer of a network is learnt for a few shot learning task. Task2Vec [146] learns a task embedding on different visual tasks which enables to predict similarities between them and how well a feature extractor perform on a chosen task. Differently from these works, we focus on learning a fixed-size embedding for diverse network architectures from which it is possible to draw ready-to-use weights for a specific

task, even for networks unseen during training.

Network Parameters Prediction. Many works deploy an auxiliary network to obtain the weights of a target network. Hypernetworks [147] trained a small network (the hypernetwork) to predict weights for a large target network on a given task. The same technique has been extended and applied in many ways: transforming noise into the weights of a target network (Bayesian setting) [148, 149], adapting the weights of a target network to the current situation [150, 151], generating weights corresponding to hyperparameters [152], focusing on the acceleration of the architecture search problem [153]. Moreover, networks that generate their own weights have been proposed and analyzed [154, 155]. While these works share the use of a weights generation module with our work, our novel proposal consists in showing how to learn a fixed-size structured embedding for different architectures and navigate through this space to obtain new weights for these kinds of architectures as well as for architectures not provided as training examples.

Weight-sharing NAS. In Weight-sharing NAS, optimal architecture search occurs over the space defined by the subnets of a large network, the supernet. Commonly, subnets share weights with the supernet and they are available as ready-to-use networks after training. OFA [156] starts by training the entire supernet and progresses considering subnets of reduced size. After training, desired subnets are selected with an evolutionary algorithm. In NAT [157], many conflicting objectives are considered, training only the weights of promising subnets for every objective. While these works deal with obtaining ready-to-use networks that obey to desired characteristics, we focus on the embeddability of deep models in a latent space organized according to features of interest and on the possibility of explore such latent space by interpolation or optimization.

5.3 Method

Framework. In the following, we will use *architecture* to denote the structure of a deep learning model (i.e. number and kind of layers, etc.) and *instance* for an architecture featuring specific parameter values. Of course, given one architecture there can be many instances with different parameter values.

Our framework, dubbed NetSpace and shown in Fig. 5.1, is able to encode trained instances of different architectures into a fixed-size encoding and to decode this embedding into new instances that behave like the input ones. The parameters of each instance presented in input to our framework are stored by a simple algorithm into a PRep (parameters representation), a 2D matrix whose rows are filled one after the other with the sequence of the unrolled parameters. Further details on the PRep generation algorithm are provided in the Appendix C.2.

NetSpace encoder takes as input the PRep of an instance and produces a small fixed-size embedding, applying first horizontal and then vertical convolutions, alongside with max-pooling. It is worth pointing out that the encoder is designed to produce embeddings of the same size for any input PRep dimension. The embedding from the encoder is then processed by NetSpace decoder, whose basic block first applies convolutions to increase the depth of the input and then reshapes the intermediate output to grow along spatial dimensions at the cost of depth. Once the required

PRep resolution has been reached, an independent linear scaling is applied to every element of the predicted PRep, with weights and biases learnt during the training. We found that this is needed, in particular, for very deep models, probably because convolutions struggle to predict parameters that are close in the PRep but that belong to distant layers of the target architecture. The values of the predicted PRep are loaded into a ready-to-use instance.

The building blocks of the encoder and decoder are specified in more details in Fig. 5.1. In the remainder of this chapter, we will use the term *target* instance to refer to the one in input to NetSpace and the term *predicted* instance to refer to that instantiated with values from the predicted PRep.

Single-Architecture Setting. In the Single-Architecture setting, NetSpace is used to learn an embedding space for the parameters of multiple instances of a single architecture. The first scenario that we consider deals with instances that exhibit different *performance* in solving the same task, such as image classification. Thus, during NetSpace training, the objective is to learn how to predict weights that match the performance of the target instances. Akin to common practice in Knowledge Distillation [158], this can be achieved by minimizing a loss term $\mathcal{L}_{pred}$ that represents the discrepancy between the outputs computed by the target instances and those computed by the corresponding predicted instances. Formally, considering a target instance N_t and training samples x with labels y, we denote by N_p the instance predicted by NetSpace when the input instance is N_t, and by $t = N_t(x)$ and $p = N_p(x)$ the logits computed by the target and predicted instances, respectively. We then realize $\mathcal{L}_{pred}$ as in [158]:

$$\mathcal{L}_{pred} = KL(\text{softmax}(p/T),\ \text{softmax}(t/T)) \cdot T^2 \tag{5.1}$$

where KL denotes the Kullback–Leibler divergence averaged across the samples. As in [158], the softmax functions used in $\mathcal{L}_{pred}$ have inputs divided by a temperature term T.

A second scenario deals with networks sharing the same architecture that are trained to fit different signals. In particular, different works [141, 142, 140, 143] have shown that it is possible to build neural representations of signals by training MLPs to regress such signals. In this scenario, each instance of the same MLP architecture is trained to represent a different signal. Given one of such instances, the objective of NetSpace is to predict weights capable of regressing the same signal. To achieve this goal, NetSpace can be trained with a loss term that directly compares the outputs of the predicted instance to those computed by the target one, thereby, also in this case, distilling the knowledge of the target instance into the predicted one. In this scenario, then, $\mathcal{L}_{pred}$ becomes simply:

$$\mathcal{L}_{pred} = MSE(y_p,\ y_t) \tag{5.2}$$

i.e. the Mean Squared Error (MSE) between the outputs from the predicted instance (y_p) and those from the target instance (y_t) when queried by the same inputs. In particular, in the experiments we consider MLPs trained to regress the Signed Distance Function (SDF) of a 3D shape (e.g., [141]).

We found that, in both scenarios, using a distillation loss is more effective than using a weights reconstruction loss, as the latter would aim just at mimicking on average the weights of the target instances, which we found not implying similar predictions. Furthermore, a distillation loss

allows for using each target instance to create many training examples for NetSpace by simply varying the input data.

Multi-Architecture Setting. In the Multi-Architecture setting, we investigate on how to embed in a common space instances having different architectures. Thus, we consider instances trained to solve an image classification task with the best performances allowed by their architecture. NetSpace is trained to process such instances and to predict weights that reproduce their good performances. In order to ease NetSpace task, we take advantage of the complete Knowledge Distillation described in [158]. Denoting by N^* a teacher network with good performances in the task at hand and by t^* its logits for a batch of images x, we define the loss with respect to it as:

$$\mathcal{L}^*_{pred} = KL(\text{softmax}(p/T),\, \text{softmax}(t^*/T)) \cdot T^2 \tag{5.3}$$

Then, as in in [158], we introduce an additional term $\mathcal{L}_{task}$ which, in combination with $\mathcal{L}^*_{pred}$, defines the complete $\mathcal{L}_{kd}$:

$$\mathcal{L}_{task} = CE(\text{softmax}(p),\, y) \tag{5.4}$$

$$\mathcal{L}_{kd} = \alpha \cdot \mathcal{L}^*_{pred} + (1 - \alpha) \cdot \mathcal{L}_{task} \tag{5.5}$$

where CE denotes the Cross Entropy loss averaged across the samples of the batch and α is a hyperparameter used to balance the two terms in $\mathcal{L}_{kd}$.

As far as the possibility of handling different architectures is concerned, we identify each architecture uniquely with a categorical ClassId. In this configuration, NetSpace is trained to predict an instance with the same architecture as the target. Even if this information is available at training time from the target instance itself, we would also like to explore by means of interpolation or optimization the latent space learnt by NetSpace after having trained it, without feeding input instances to the framework. Thus, we wish to be able to extract the architecture information directly from the embedding. To achieve this objective, we modify the architecture presented so far by adding a softmax classifier on top of the embedding in order to predict the ClassId of the target instance (details on this variant of the framework are reported in the Appendix C.1). Consequently, we complement the learning objective introduced in Eq. 5.5 with an additional $\mathcal{L}_{class}$ term. Given a target instance N_t and the embedding e produced by NetSpace encoder for it, we denote by c_t the ClassId associated to the architecture of N_t and by c_p the logits predicted by the architecture classifier from e. $\mathcal{L}_{class}$ is then defined as the Cross Entropy loss between the predicted and target ClassId:

$$\mathcal{L}_{class} = CE(\text{softmax}(c_p),\, c_t). \tag{5.6}$$

The initial experimental results highlighted that NetSpace was clustering the latent space according to the architecture ClassId only. We judge such organization of the embedding space as not satisfactory, as it would allow, perhaps, to sample new instances within a cluster by proximity or interpolation, but there would be no simple technique to navigate from one cluster to the others. Rather, we aim at endowing the embedding space with a structure enabling exploration

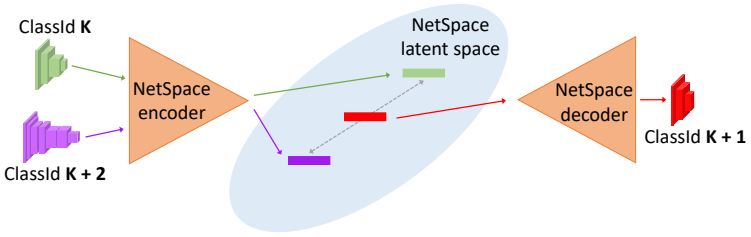


FIGURE 5.2: An example of interpolation in the latent space learnt by NetSpace in the Multi-Architecture scenario.

along meaningful directions, i.e. directions somehow correlated to a specific characteristic, such as number of parameters or performance. Thus, a more amenable organization would consist in clusters showing up *aligned*, rather than scattered throughout the space, and possibly also *sorted* w.r.t. a given characteristic of interest.

Should such organization of the embedding space be possible, given two *boundary* embeddings (i.e. representing two instances with the smallest and the largest value of the characteristic of interest), it could be possible to move across the aligned clusters by simply interpolating the boundaries and obtain along the way representations of ready-to-use instances with increasing values of the characteristic of interest. To further investigate along this path, we shall consider first that it is possible to assign ClassIds to a pool of architectures so as to sort them accordingly to a characteristic of interest. For instance, in our experiments, ClassIds K and $K+1$ will denote two architectures such that the latter has more parameters than the former.

Therefore, we introduce a new loss, denoted as $\mathcal{L}^{\gamma}$ (Interpolation Loss), whose objective is to impose the desired ordered alignment of clusters in the latent space. Given training instances belonging to boundary architectures (i.e. those with the smallest and largest ClassId), we first use NetSpace encoder to obtain their embeddings. Then, we interpolate such embeddings according to a given factor γ, and constrain the interpolated embedding to belong to the architecture whose ClassId is interpolated between the boundaries according to the same factor γ. Fig. 5.2 presents an example of the interpolation procedure described in this paragraph.

Formally, given boundary embeddings e^A and e^B of target instances N_t^A and N_t^B with ClassIds c^A and c^B, we define the interpolated embedding $e^{\gamma} = (1-\gamma)\cdot e^A + \gamma \cdot e^B$. Then, considering the logits c_p^{γ} predicted by the ClassId classifier for e^{γ} and the interpolated ClassId $c_t^{\gamma} = (1-\gamma)\cdot c^A + \gamma \cdot c^B$, we define $\mathcal{L}_{class}^{\gamma}$ to impose the consistency of the interpolation factor for ClassId as:

$$\mathcal{L}_{class}^{\gamma} = CE(\text{softmax}(c_p^{\gamma}),\ c_t^{\gamma}). \tag{5.7}$$

Moreover, considering the instance N_p^{γ} predicted by NetSpace from e^{γ}, we denote by p^{γ} the logits predicted by such instance for a batch of images and define $\mathcal{L}_{kd}^{\gamma}$ as:

$$\mathcal{L}_{pred}^{\gamma} = KL(\text{softmax}(p^{\gamma}/T),\ \text{softmax}(t^*/T)) \cdot T^2 \tag{5.8}$$

$$\mathcal{L}_{task}^{\gamma} = CE(\text{softmax}(p^{\gamma}),\ y) \tag{5.9}$$

$$\mathcal{L}_{kd}^{\gamma} = \alpha \cdot \mathcal{L}_{pred}^{\gamma} + (1-\alpha)\cdot \mathcal{L}_{task}^{\gamma} \tag{5.10}$$

with the objective of distilling the teacher network N^* also in the interpolated instances. Finally, we define the total interpolation $\mathcal{L}^\gamma$ as:

$$\mathcal{L}^\gamma = \mathcal{L}^\gamma_{class} + \mathcal{L}^\gamma_{kd} \tag{5.11}$$

In our framework, we use $\mathcal{L}^\gamma$ with different interpolation factors γ, whose values are computed according to the number of considered architectures. More precisely, considering A architectures, γ can be computed as:

$$\gamma = \frac{i}{A-1} \quad i \in \{1, 2, ..., A-2\}. \tag{5.12}$$

Given a batch of instances, we compute $\mathcal{L}_{kd}$ and $\mathcal{L}_{class}$ on each of them. Then, we apply $\mathcal{L}^\gamma$, with γ values obtained from Eq. 5.12, on all the pairs composed of instances with the minimum and maximum ClassId. The final loss, thus, is the sum of $\mathcal{L}_{kd}$, $\mathcal{L}_{class}$ for each instance of the batch and $\mathcal{L}^\gamma$ for each pair of boundary instances.

5.4 Experiments

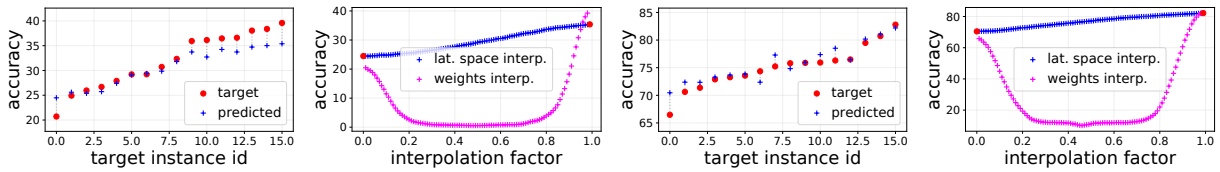
We test our framework with networks trained on image classification and 3D SDF regression.

Datasets and Architectures. For what concerns image classification, we report results on Tiny-ImageNet (TIN) [159] and CIFAR-10 [160] datasets. The target architectures for our experiments are LeNetLike, a slightly modified version of the lightweight CNN introduced in [161], VanillaCNN, a sequence of standard convolutions followed by a fully connected layer, and two variants of ResNet [40], namely ResNet8, and ResNet32.

As far as 3D SDF regression is concerned, we consider MLPs trained to overfit a selection of ~ 1000 *chairs* from the Shapenet dataset [162]. Each MLP has a single hidden layer with 256 nodes and uses periodic activation functions as proposed in [140]. Additional details are available in Appendix C.3, C.4, and C.5.

Single-Architecture Image Classification. As a first experiment, we test NetSpace in the Single-Architecture setting with the image classification task on CIFAR-10 and TIN. We create a dataset of 132 randomly initialized ResNet8 instances, training them for a different numbers of epochs, to collect instances with different performances. Then we randomly select 100 instances for training, 16 for validation, and 16 for testing.

Fig. 5.3a and 5.3c compare the accuracy achieved on TIN and CIFAR-10 test sets by target and predicted instances. The target instances belong to the test sets and were never seen by NetSpace at training time. We can see that, beside few outliers, our framework is effective in predicting new instances that emulate the behavior of the target ones, both on CIFAR10 and on the larger and more varied TIN. It is remarkable that NetSpace is able to reconstruct an instance which follows the input one in terms of performance in spite of the huge compression it introduces. Indeed, the embedding size is a fraction of the number of parameters of the instances it can reconstruct, e.g. it is 4096 for TIN, only $\sim 3.18\%$ of the parameters of ResNet8. The key information about



(A) TIN test set (B) TIN interp. (C) CIFAR10 test set (D) CIFAR10 interp.

FIGURE 5.3: Single-Architecture results for ResNet8. (a) and (c): Accuracy achieved on the test set by target and predicted instances. Target instances are sorted w.r.t. their performances on the test set. (b) and (d): Accuracy achieved on the test set by instances predicted from interpolated embeddings.

the behaviour of a neural net seems to live in a low dimensional space. Indeed, as shown by the visualization of PReps provided in Appendix C.7, the predicted instance is very different from the target one: NetSpace captures the essential information to reproduce the behaviour of the target network, it does not merely learn to reproduce it.

After training, NetSpace has learnt to map target instances to fixed-sized embeddings. Thus, we can use NetSpace frozen encoder to obtain the embeddings of two anchor instances and linearly interpolate between them in order to study the representations laying in the space between the two anchors. To this aim, we decode every interpolated embedding to generate a new instance with NetSpace frozen decoder and compute the accuracy of this ready-to-use instances on the images in the test sets. As a baseline, we consider the possibility of interpolating directly the weights of the anchor instances. Results are reported in Fig. 5.3b and 5.3d for TIN and CIFAR-10, respectively.

Interestingly, along the multi-dimensional line connecting the anchor embeddings we find representations corresponding to instances whose performances grow almost linearly with the interpolation factor, while interpolating directly the weights of the anchors yields random performances almost everywhere. This result suggests that the embedding space learnt by our framework can be organized to have meaningful dimensions, that are not exhibited in the instances weights space. In fact, the loss function used in the Single-Architecture training concerns performance and our framework learns *naturally* a latent space that, at least locally, can be explored along a direction strictly correlated with performance.

Single-Architecture SDF regression. As a second Single-Architecture experiment, we train NetSpace to learn a latent space of MLPs that represent implicitly the SDF of chairs from the ShapeNet dataset. We train our framework on a dataset of $\sim$ 1000 MLPs: each of them has been trained to overfit a different 3D shape, starting from a different random initialization. The goal of this experiment is to assess if NetSpace is capable of learning a meaningful embedding of 3D shapes, which can then be explored by linear interpolation. Thus, after training NetSpace, we obtain two anchor embeddings by processing two input MLPs with NetSpace frozen encoder. Then, we obtain new embeddings by interpolating the anchors and we predict new MLPs with NetSpace frozen decoder. The results of this experiment are reported in Fig. 5.4. The top two rows show interpolation results obtained from NetSpace latent space, while the bottom row presents results obtained by interpolating directly the weights of the anchor MLPs.

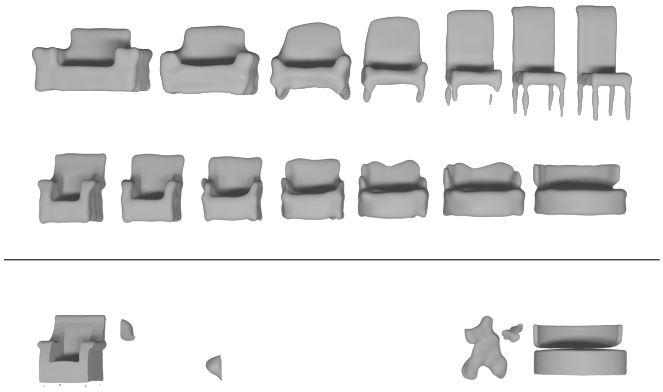


FIGURE 5.4: Interpolation of 3D shapes. Top two rows: results obtained by interpolating NetSpace embedding space. Bottom row: the same linear interpolation applied to MLPs weights.

We can notice that direct interpolation in the MLPs weights space yields catastrophic failures, while NetSpace embedding space enables smooth interpolations between the boundary shapes. This shows its ability to distill the core content of a trained model into a small-size embedding abstracting from the specific values of weights, and also its flexibility: when the loss concerns fitting of shapes, the latent space of models *naturally* organizes to have dimensions correlated with shape.

Multi-Architecture. In the Multi-Architecture setting we train NetSpace to embed four architectures: LeNetLike, VanillaCNN, ResNet8 and ResNet32. To build the dataset, we train many randomly initialized instances for each architecture, collecting multiple instances with good performances (100 for training, 16 for validation and 16 for testing). We collect in total 400 instances for training, 64 for validation and 64 for testing. We adopt a ResNet56 with high performance as the teacher network in Eq. 5.3 and 5.11 and set α to 0.9 in Eq. 5.5.

We observe that supervision is not the same for different architectures: instances with lower performances receive a stronger signal from the $\mathcal{L}_{kd}$ and $\mathcal{L}^{\gamma}$ provides additional supervision for non-boundary architectures. We alleviate this issue modifing the training set so as to include a different number of instances for each architecture : we include 60 LeNetLike, 50 VanillaCNN, 60 ResNet8 and 100 ResNet32 instances.

Fig. 5.5 left shows the results of this experiment: NetSpace successfully embeds instances of multiple architectures in highly compressed representations with all the information needed to a) predict correctly the architecture of the target instance and b) reconstruct an instance of such architecture whose behavior mimics that of the target one.

Multi-Architecture Embedding Interpolation. As we defined the ClassIds of architectures according to their increasing number of parameters, we expect their latent representations to be sorted w.r.t. this characteristic thanks to our interpolation loss $\mathcal{L}^{\gamma}$. In this experiment, we explore the latent space by observing the classification accuracy achieved by instances obtained when interpolating one embedding of LeNetLike and one of ResNet32, while moving with smaller steps than those defined in Eq. 5.12.

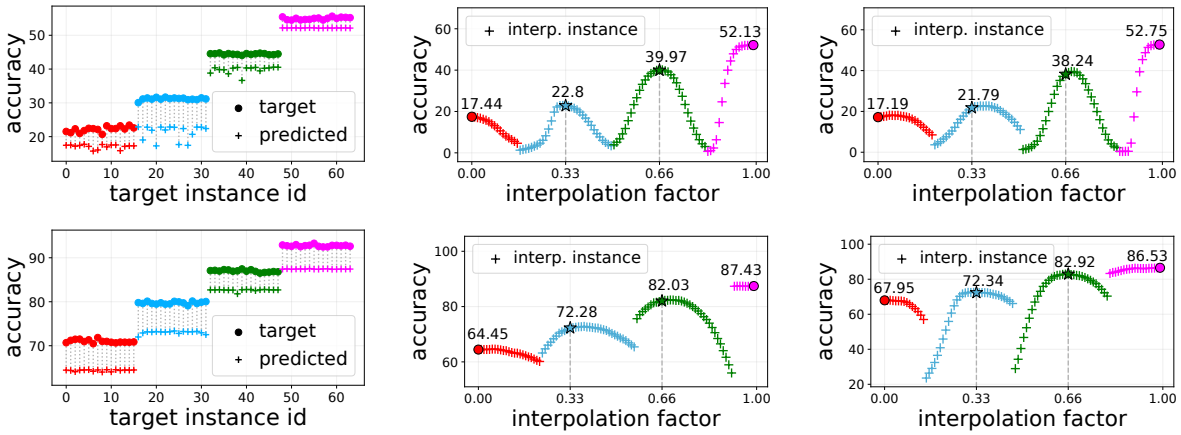


FIGURE 5.5: Results of the Multi-Architecture setting on TIN (top) and CIFAR10 (bottom). Left: target instances from test set, instances are sorted w.r.t. their ClassId. Center: interpolation with all architectures available at training time. Right: interpolation with only variants of LeNetLike and ResNet32 seen at training time. In all figures, color represent architecture: red-LeNetLike, blue-VanillaCNN, green-ResNet8, fuchsia-ResNet32. Circles correspond to interpolation boundaries. Stars denote instances obtained with the discrete interpolation factors used in $\mathcal{L}^{\gamma}$.

Notably, as shown in Fig. 5.5 center, NetSpace learns an embedding space where architectures vary according to their number of parameters along an hyper-line. Moreover, it organized the space to place best performing embeddings for every class around the positions on which $\mathcal{L}^{\gamma}$ was computed.

Sampling of Unseen Architectures. We perform a new Multi-Architecture experiment by using a training set composed only of LeNetLike and ResNet32 instances. We collect 40 instances for LeNetLike and 80 for ResNet32, with the same balancing strategy discussed above. By not showing to NetSpace encoder any instance of VanillaCNN and ResNet8, we deny it the possibility to learn directly a portion of the embedding space dedicated to them. However, $\mathcal{L}^{\gamma}$ shapes it indirectly: for instance, given two embeddings e_{lenet} and e_{r32} of, respectively, a LeNetLike and ResNet32, it forces the embedding $e^{\gamma} = 0.33 \cdot e_{lenet} + 0.66 \cdot e_{r32}$ to represent an instance of ResNet8 (unseen during training).

After training, we perform an interpolation experiment and report results in Fig. 5.5 right: we find that the latent space learnt by NetSpace trained on a reduced set of architectures exhibits the same properties as the space learnt by training with all of them, allowing to draw by interpolation instances with good performance of the *unseen architectures* VanillaCNN and ResNet8.

Latent Space Optimization. Here we investigate the navigation of NetSpace embedding by latent space optimization (LSO). In order to do so, we take NetSpace encoder and decoder obtained by a Single-Architecture Image classification experiment, freezing their parameters. Then, we obtain an initial latent code by embedding one ResNet8 instance from the test set with the frozen encoder. We then use the frozen decoder to perform an iterative optimization of the initial embedding. In each step of the optimization, the embedding is processed by the frozen decoder, which predicts a PRep that is loaded into a ResNet8 instance. The resulting network is then used

	Single-Architecture	Multi-Architecture			
	ResNet8	LeNetLike	VanillaCNN	ResNet8	ResNet32
initial	25.73%	17.44%	22.89%	38.81%	52.17%
optimized	35.72%	18.55%	24.17%	42.07%	53.13%

TABLE 5.1: Accuracy on TIN test set achieved with LSO.

to produce predictions on a batch of training images from TIN. We apply $\mathcal{L}_{kd}$ on these predictions using a ResNet56 as teacher network, to guide NetSpace in the search of a high performing instance in the learnt latent space. As the decoder is frozen, we compute the gradient of the loss w.r.t. the embedding, so as to explore the latent space by gradient descent.

Furthermore, we perform a similar experiment starting from NetSpace encoder and decoder taken from a Multi-Architecture training experiment. Also in this case, we use the frozen encoder to obtain initial embeddings from four instances belonging to different architectures. Then, we process each embeddings with the frozen decoder, which predicts a PRep and a ClassId. We use the predicted ClassId to select the architecture where the predicted PRep is loaded, building an instance which we use to produce predictions on a batch of training images from TIN. In addition to $\mathcal{L}_{kd}$, in this case we apply also $\mathcal{L}_{class}$ on the predicted ClassId, to guide the embedding towards the area of the latent space that corresponds to the desired class. In Table 5.1 we report the performances obtained by the optimizations (second row), together with the performances of the instances used to obtain the initial embeddings (first row).

Remarkably, the results show that it is actually possible to improve the performances of the input instances by exploring the NetSpace embedding space via latent space optimization.

5.5 Conclusions and Future Work

NetSpace introduces a framework to learn the latent space of deep models. We have shown that the embedding space learnt by NetSpace can be organized according to meaningful traits and ready-to-use instances with predictable properties can be obtained by means of linear interpolation or latent space optimization. Furthermore, our experiments provide evidence that fixed-size embeddings can represent effectively instances of several different architectures. The main limitation of our work concerns lack of investigation on the applications of our findings, but we believe that being able to create low-dimensional representations of complex models organized according to user-defined traits may open up a broad range of possibilities for further studies, regarding, for example, network compression or interpolation for transfer learning. Therefore, we deem it worth communicating our findings to the community as we believe they are non-obvious and valuable on their own, and may foster further research as well as identification of the most fruitful outcomes toward applications.

Chapter 6

Conclusions

This thesis investigates the problem of gathering geometric information when dealing with non-professional freehand sketches, an imprecise, abstract but powerful way to represent realistic and imaginary objects and scenes. Two main topics were addressed, the generation of geometric information from freehand scene sketches to improve 2D sketch-based tasks and the use of Vision-Language models to overcome sketch-based 3D tasks limits. Novel approaches were presented achieving interesting results and showing promising future directions to further progress this research line.

Regarding the first topic, SketchyDepth is presented, a framework able to generate an RGB image and the correspondent depth map from a freehand scene sketch. SketchyDepth achieves improved image generation performance and unlocks new creative effects exploiting the geometric information of the generated depth map.

This work also addressed Vision-Language models applied to sketch-based 3D tasks. A study to analyze the effectiveness of a perceptual loss defined on the CLIP embedding space to discriminate objects and views between different sketch related domains is conducted. Both settings achieved good experimental results, matching objects between different image domains have shown good performance, the same goes for objects views. Motivated by this study, CLIP has been applied to sketch-based 3D shape retrieval, proposing a solution based on two main contributions. A methodology to study relevant factors to the retrieval task and find the best setting and a learned technique to mask or weight non-relevant CLIP features for a specific downstream task. This approach achieved a new state-of-the-art performance without the need of training on supervised data. The investigation on Vision-Language models is further extended to the freehand sketch to 3D reconstruction task. An optimization procedure for the inversion of a pre-trained 3D StyleGAN from freehand sketches is defined. A comparison between the CLIP embeddings of target sketches and generated shapes renderings is implemented to guide the inversion. The obtained shapes are not always well constructed but they show some features that resemble the objects depicted in the target sketches, a promising result that motivates further research.

Overall, this work was able to investigate the association of geometric information to freehand sketches with positive results, following novel directions and improving over previous work from multiple perspectives.

This work also paves the way for future research, both extending the covered topics and defining novel directions that are still connected with the presented subjects. Extending the obtained results on sketch-based 3D shape retrieval so as to generalize to many more classes without the

need of supervised data would have an important impact on the task. It would make it possible to retrieve with state-of-the-art performance existing 3D models simply by sketching objects with one unique general solution. Showing fine-grained retrieval results, i.e. discriminating different objects of the same class, will require the acquisition of a new test set dataset with {freehand sketch, 3D shape} pairs. This is still affordable because only a limited amount of data to validate and test our method is needed, the training of an entire network is not needed.

The application of Vision-Language models for unsupervised 3D reconstruction is a straightforward direction for future research. This is the most recent topic of this research work, results are promising and, as reported in Section 4.3.4, there are many ways to progress from the current approach.

A novel and interesting research direction would instead be to combine the expressive power of the freehand sketch with that of the natural language to guide 3D reconstruction, taking full advantage of Vision-Language models. The two input conditioning would complete each other, a sketch would convey information like the position and details of objects, that are very difficult to describe with text, while text could describe the style and content of a desirable background with just a few words, something that, using sketching, could require a lot of time, effort and eventually skill.

In the last chapter also a different research topic has been addressed, regarding learning of the embedding space of trained neural networks that can be navigated by interpolation or optimization to get ready-to-use networks with desired characteristics. We believe that NetSpace is the spark of a novel research direction and that could represent a starting point for other researchers willing to explore an exciting new way to deal with neural networks.

Appendix A

Inferring Depth from Scene Sketches

FIGURE A.1: Left: generated 256x256 image from a test sketch. Right: small translations applied on the same sketch to improve diversity [57]

A.1 Additional Qualitative Results

A.1.1 Depth Based Creative Effects

As an extension of Fig. 3.5, in Fig. A.2 we show additional examples of depth-based effects obtained with our generated images and depth maps on sketches belonging to the test set.

A.1.2 Colour-to-Grayscale and Transition-to-Cartoon Effects

Fig. A.3 reports examples of two additional effects attainable by leveraging on the depth maps generated alongside images, namely colour-to-grayscale and transition-to-cartoon. Both effects are applied to each pixel of a generated image based on the corresponding generated depth. The colour-to-grayscale effect keeps near pixels almost unaltered while at increasing distances pixels progressively loose colour. The transition-to-cartoon instead applies a strong cartoon effect for near pixels, reducing it progressively with increasing distance values.

The cartoon overlay image used for the cartoon effect is obtained starting from a generated one by a sequence of image processing operations. First of all, we transform the image to grayscale, blur it by a median filter and apply an adaptive thresholding operation to extract edges. Then we smooth the original image by applying a bilateral filter and highlight the extracted edges by colouring them in black.

A.1.3 Depth Based Effects: our approach vs. MiDaS

In Fig. A.4, we show depth-based effects, i.e. Bokeh, light variation and fog, obtained by using either our depth maps, which are generated jointly with images, or the depth maps predicted by MiDaS [108] on the generated images.

As for the Bokeh effect, we consider a fixed depth value to determine where the blur filter starts to affect the image. This parameter represents a distance threshold to separate foreground from background and it is kept fixed between the depth maps yielded by our method and MiDaS, as well as throughout all the presented examples. Then, starting at the defined distance threshold, Bokeh progressively increases image blur as the depth gets higher. All the other effects are applied using the same parameter values as in the examples shown in chapter 3. Light variation decreases brightness of closer pixels with respect to those farther away. Conversely, the fog effect renders more foggy the pixels exhibiting larger depths.

Figure A.4 consists of three pairs of rows. In each pair, the top row depicts the results dealing with our method, the bottom one those concerning MiDaS. Every row reports, from left to right, a generated image, the associated depth map, the Bokeh, light variation and fog effects.

In the first pair of rows, we note the clear difference between our generated depth map, that looks amenable to realize effects based on discriminating between foreground, in this case a giraffe, and background, and the depth map yielded by MiDaS, that seems to fail in separating foreground from background neatly. In our results (top row), the relighting effect darkens the whole giraffe evenly, while the background is clearly brighter. In the bottom row, instead, we can see how the relighting based on the depth map computed by MiDaS fails to darken differently the different portions of the image, i.e. the giraffe and the background exhibit similar brightness.

Similarly, we can see how our depth map allows for simulating a foggy background, while the effect based on MiDaS does apply the fog to both the background as well as the giraffe. The Bokeh filter seems also rather dependent on the quality of the depth map. In the bottom row (MiDaS) the head and neck of the giraffe are blurred similarly to the background, while our depth map (top row) allows for simulating much more effectively a shallow depth of field, with the foreground object looking sharper than the background.

Similar considerations can be drawn from the analysis of the third pair of rows, due to, again, the much more accurate foreground-background separation achieved by our generated depth map compared to that computed by MiDaS. Indeed, from left to right, in our results (top row) the zebra in the center of the image turns out much sharper, darker and less foggy than the background, whilst this is definitely not the case in the bottom row (MiDaS).

Finally, also in the second pair of rows we can see how our depth maps are conducive to nice creative effects while those applied based on MiDaS show several issues. In fact, in the bottom row the Bokeh and fog filters are almost ineffective due to the whole image but the sky being treated as foreground. Accordingly, the light variation obscures most of the image, making, again, nearly no difference between foreground objects and the background.

Overall, our experimental findings show that, most of the times, the depth map computed by MiDaS on top of an image generated from a sketch is significantly less amenable to support depth-based creative filters than the depth map we can generate jointly together with the corresponding image by the proposed network architecture and training procedure.

A.1.4 Image and Depth Map Generation Examples

To provide a more comprehensive collection of qualitative results, in Figures A.5 and A.6 we report additional comparisons between our generated scene images and depth maps and the corresponding baselines.

A.1.5 Depth Map Sketching Examples

As an extension to figure 3.7, in figure A.7 we show further examples of scene image generation through the novel depth sketching approach peculiarly enabled by our proposal.

A.1.6 Image Generation Resolution and Diversity.

We experimented image generation with different resolution keeping the foreground generation part fixed and training the rest of the system at 256x256 resolution. In Fig. A.1 left is visible a generated test example.

For what concern diversity in image generation, we keeps dropout active also at test time, following [97]. Here we show that also "conditioning perturbation" [57], implemented as small translations on the objects of the input sketch, can be deployed in our method to increase diversity, as shown in Figure A.1 right.

FIGURE A.2: Examples of creative effects enabled by our generated depth maps. From left to right: generated image, Bokeh, light variation, fog and hue shift.

FIGURE A.3: Colour-to-grayscale and transition-to-cartoon effects. From left to right: generated image, colour-to grayscale, transition-to-cartoon.

FIGURE A.4: Comparison between depth-based effects obtained by our generated depth maps and MiDaS depth maps. Results are subdivided in three pairs of rows. In each pair, the top row deals with our depth map, the bottom one concerns MiDaS. Every row displays, from left to the right, a generated image, its depth map, the Bokeh, light variation and fog effects. See text for comments.

FIGURE A.5: Comparison between our generated images and baseline (our replication of SketchyCOCO [62]) results. In both columns, from left to the right: scene sketch, image generated by our method, image generated by the baseline method.

FIGURE A.6: Comparison between the depth maps generated by our method and those obtained by MiDaS. In both columns, from left to right: our generated image, our generated depth map and the depth map computed by MiDaS from our generated image.

FIGURE A.7: Image generation by depth sketching. From left to right: generated depth map, generated image, sketched depth map and newly generated image.

Appendix B

CLIP-facilitated sketch-based 3D shape retrieval

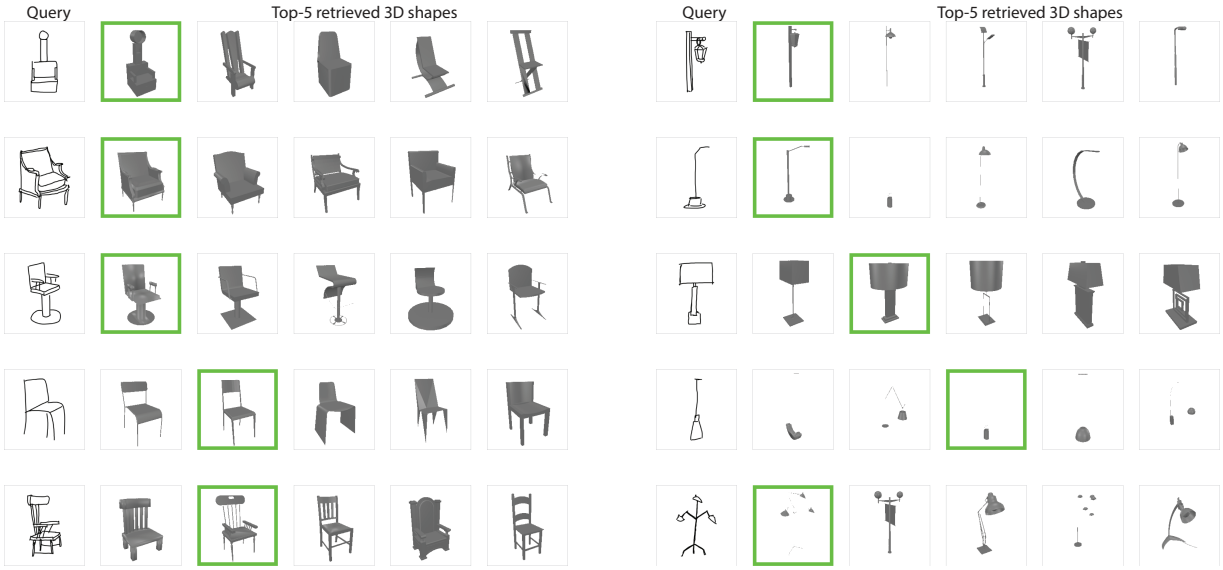


FIGURE B.1: Additional qualitative results of our full method (including object scaling and centering to fit the optimal bounding box, feature computation from several augmented views versions and features masking). Left: examples from the chair category; Right: examples from the lamp category. In both cases, the first column shows the query freehand sketch and the other columns show the top-5 retrieved 3D shapes. We render 3D shapes using textureless RGB rendering, with camera azimuth angle set to 30° and zenith angle to 10°. Freehand sketches with different viewpoint of the same object are considered as different queries.

All the experiments in this appendix share the following settings: freehand sketches are used as queries, and the gallery 3D shapes are represented with their multi-view line renderings; all the results are evaluated on the test sets.

B.1 Additional qualitative results & observations

In Fig. B.1, we show additional qualitative results of our method for both, chairs and lamps. As for Fig. 4.4, we *randomly* select retrieval examples from the test set of sketch-shape pairs for chair and lamp categories. In the test set, the 3D shapes gallery consists of 201 chair models and 111

lamp models.

Observations Qualitative results match quantitative ones: the first matched to a query sketch 3D model is often the correct one, and when it is not, the correct match still appears in the first five matches. Additionally, the qualitative results show that *the retrieved top-5 results are visually similar to the query*.

B.2 How often does our model correctly match viewpoints of a 3D model to a query viewpoint?

In all experiments for our retrieval model, to compute the top-1 and top-5 retrieval accuracy, we simply evaluate the percentage of times the ground-truth object is among the top-1 and top-5 ranked retrieval results. In this case, the distance to an object is the minimum distance from the query to object views. In this section, to compute the top-1 and top-5 retrieval accuracy, we rank distances to individual views of each gallery object. In this case, our top-5 ranked retrieval result can contain the same object multiple times, e.g.: <Object n, view i>, <Object n, view j>, <Object k, view i> ...

Table B.1 shows that the new top-1 retrieval accuracy values are lower by around 15 points for the chair category and around 27 points for lamp category. This performance gap shows that even though using distances in the CLIP latent space we might not be able to identify a matching shape viewpoint, CLIP tends to project viewpoints of the ground-truth object closer to a query than viewpoints of other objects.

The difference in top-1 accuracy (according to the new accuracy calculation criteria) for the lamp and chair categories can be explained by the fact that many lamps look exactly the same from different viewpoints.

The top-5 retrieval accuracy drops around 5 points for the chair category and 4 points for the lamp category, showing that our model is capable to select the exact view among the first five choices most of the times.

	Chair		Lamp	
	acc@1	acc@5	acc@1	acc@5
our, correctly selected object	75.95	88.39	74.17	88.59
our, correctly selected view	59.70	82.59	47.15	84.08

TABLE B.1: Correctly selected views.

B.3 Ablation studies

B.3.1 Additional discussion on objects scaling and centering in sketch and model views.

In this section, we provide additional ablation on the choice of object centering, scaling in sketch and model views.

Baseline *Baseline* method in Table 4.8, uses the freehand sketches from the dataset by Qi et al. [5], which we center and scale to fit the bounding box of size 170×170. This corresponds to the smallest square bounding box that fully encompasses objects in the original sketches. The line renderings are obtained as described in the beginning of Section 4.2.3.

The role of objects centering in retrieval accuracy In our method we propose to center and scale objects in sketch and model views to fit the the optimal bounding box (OBB). In Section 4.2.3.2, we only consider the role of the chosen scale of the bounding box. Here, we additionally evaluate the role of objects centering. Table B.2 shows that centering operations increases the performances on chairs and lamps for both metrics. For the reference, we also provide the results of additional scaling of the objects to fit the optimal bounding box (OBB).

	Chair		Lamp	
	acc@1	acc@5	acc@1	acc@5
Baseline	51.58	70.81	54.65	73.57
Baseline w/ centering	57.71	75.62	57.96	74.77
OBB	65.17	82.09	70.27	82.58

TABLE B.2: Role of objects centering in sketch and model views.

B.3.2 Contribution of each individual design choice towards our full method

In this section, we ablate contribution of each individual design choice towards our full method, which includes object scaling and centering to fit the optimal bounding box (*OBB*), feature computation from several augmented views versions (*Augm. 4 imgs.*) and features masking (*Binary*). The retrieval accuracy of our full method is shown in Table B.3 (line 1).

Table B.3 (*Cent. BB:* 170×170, *Augm. (4 imgs), Binary*) correspond to the full method, with not optimal scaling of the bounding box. Table B.3 (*OBB, No augm., Binary (4 imgs)*) corresponds to the full method where the features are computed without taking into consideration augmented versions of the views. Table B.3 (*OBB, Augm. (4 imgs), no masking*) corresponds to the full method where we do not use feature masking strategy.

Removing any of the components results in a performance drop, showing that all the components of our method are necessary to reach the best solution.

	Chair		Lamp	
	acc@1	acc@5	acc@1	acc@5
Our - OBB, Augm. (4 imgs), Binary	**75.95**	**88.39**	**74.17**	**88.59**
Cent. BB: 170 × 170, Augm. (4 imgs), Binary	64.87	80.18	66.33	82.75
OBB, No augm., Binary (4 imgs)	69.82	84.58	70.87	84.98
OBB, Augm. (4 imgs), no masking	70.64	85.24	**74.17**	86.49

TABLE B.3: Ablation study of our final solution. The checkpoint for feature masking is selected based on the performance on a validations set of freehand chair sketches.

B.3.3 Multiple images augmentation ablation study

We perform an extended study of the augmentation operation with four images introduced in Section 4.2.3.2 and Table 4.2. We evaluate the performances of all the 3-image augmentation configurations that can be obtained by removing one image at a time from the 4-image augmentation method. We show the results in Table B.4. We observe that removing the starting image leads to the most severe drop in performance. This can be explained by the fact that the settings (position and scaling) for this image correspond to the settings that we found to result in best retrieval performance. Results for other configurations show performance increases and decreases on individual categories. However, the feature computation with 4 images strikes the best balance in performance on both classes, according to both top-1 and top-5 metrics.

	Chair		Lamp	
	acc@1	acc@5	acc@1	acc@5
OBB, Augm. w/o original img. (3 imgs)	66.83	83.75	71.47	84.69
OBB, Augm. w/o tranls. (3 imgs)	70.15	84.74	**74.17**	**87.09**
OBB, Augm. w/o rot. (3 imgs)	70.32	86.23	73.57	85.89
OBB, Augm. w/o transl & rot. (3 imgs)	69.49	**87.74**	72.37	85.89
OBB, Augm. (4 imgs)	**70.64**	85.24	**74.17**	86.49

TABLE B.4: Ablation study for the four images augmentation.

B.4 Comparison of our 3D model viewpoints selection with a set of the following views: front, back and 3/4 bird eye view.

We compare the gallery views used by Schlachte et al. [124] for sketch based 3D shape retrieval with the views that humans most likely use to sketch objects of a given category. We conduct this study to better understand which configuration best suite the sketch based 3D shape retrieval task. We consider two classes, chairs and lamps, and we define as views for Schlachte et al. [124] the orthographic 0° azimuth view, the orthographic 180° azimuth view and the perspective 30° azimuth view, following the indications in the paper. The human-preferred set of viewpoints consist of the 0° azimuth, 30° azimuth and 75° azimuth perspective views for chairs; and 0° azimuth, 45° azimuth and 90° azimuth perspective views for lamps, following [5]. For the human-preferred viewpoints the elevation angle is set to 20°.

We scale objects in all viewpoints (sketch, human-preferred line rendering viewpoints, and viewpoints used by Schlachte et al. [124]) to fit OBB. We compare the performance under two

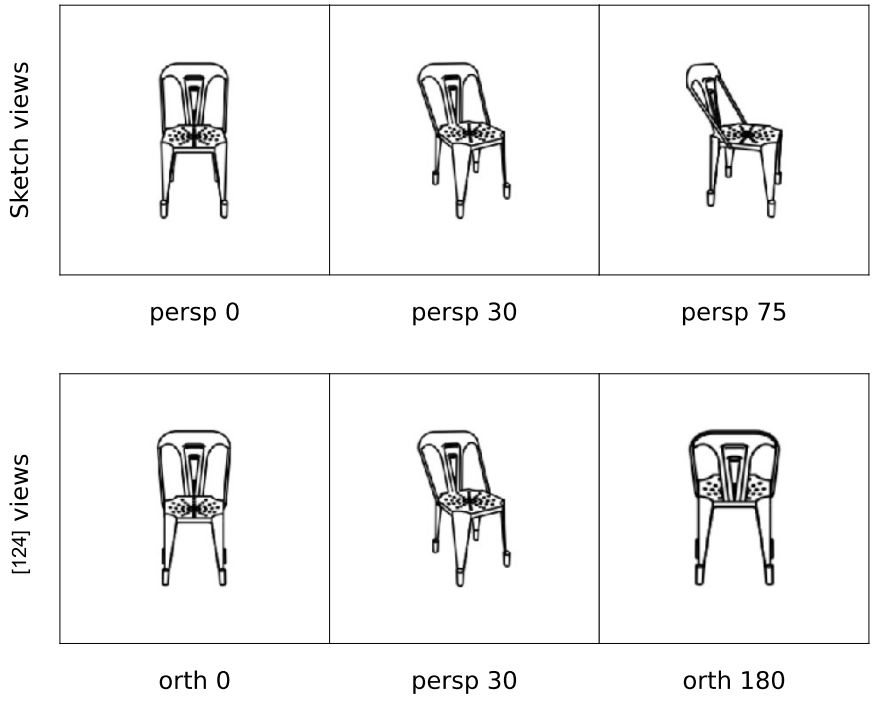


FIGURE B.2: Comparison between human-preferred sketch views and views used by Schlachte et al. [124].

feature aggregation scenarios (Section 4.2.3.2, "Minimum vs average"): as proposed in Eq. (4.1) (*min*) and averaging features from all views (*avg*).

From Table B.5, we see that in both settings, using the human-preferred viewpoints leads to better performance for both classes. In Fig. B.2, we show example line rendering viewpoints used by Schlachte et al. [124] and the human-preferred viewpoints [5].

	Chair		Lamp	
	acc@1	acc@5	acc@1	acc@5
OBB, sketch views, min	65.51	82.59	69.67	85.29
OBB, 0 (o), 30 (p), 180(o), min	50.91	72.47	53.15	78.69
OBB, sketch views, avg	59.70	79.77	55.56	75.98
OBB, 0 (o), 30 (p), 180(o), avg	46.93	70.81	44.14	68.47

TABLE B.5: Comparison between different view choices.

B.5 Additional details about learning CLIP feature masking or weighting

In all our experiments, we use a batch size of 32 and a learning rate of 0.001. As CLIP embeddings, we use the output of the third layer of the ResNet101 image encoder: it has 200704 values. The vector mask that we learn has the same number of weights. We augment images in every batch with a random affine transformation, randomly sampling translation, rotation, and scaling parameters. The translation moves an image along x and y axes for a random number of pixels between [-10%, +10%] of the image size. The rotation is sampled between [-10, +10] degrees and

the scaling increase or decrease the bonding box by a random value in the range [-10%, +10%] of the image size. The image size is set to 224x224. We choose the best checkpoints for chairs and lamps on the validation sets, considering the best retrieval score obtained with our 4 images augmented OBB setting.

Appendix C

Learning the Space of Deep Models

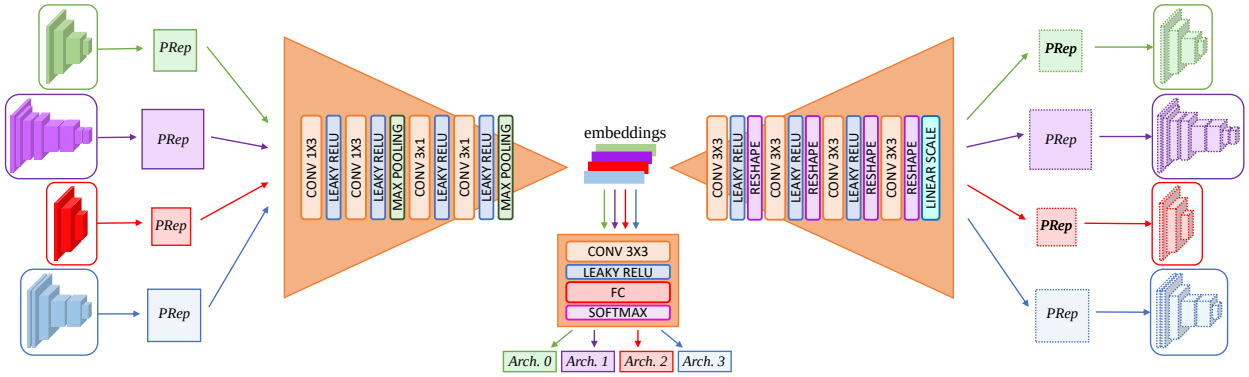


FIGURE C.1: Overview of NetSpace architecture used in the Multi-Architecture experiments.

C.1 ClassId Classifier

In Section 5.3 (*Multi-Architecture Setting*) we introduce the need to extend NetSpace architecture to the Multi-Architecture setting. In this setting, in fact, we ask our framework to extract the ClassId of the predicted instances from the embeddings. To achieve this goal, we extend the architecture of our framework with a softmax classifier, which takes in input the embeddings generated by NetSpace encoder and is trained to predict the correct ClassId with $\mathcal{L}_{class}$. The classifier is a lightweight neural network, composed of one convolutional layer and one fully connected layer, interleaved by the LeakyReLU activation function. The outputs of the classifier are transformed into probabilities by the softmax function. Fig. C.1 presents an overview of the version of NetSpace used in the Multi-Architecture case, including the ClassId classifier.

C.2 Parameters Representation (PRep)

A PRep is a 2D tensor where we store all the parameters of an instance of a neural network by means of a simple algorithm exemplified in Fig. C.2.

We designed our framework fixing PReps to be rectangular matrices with high width/height ratio. Additionally, to favor easy implementation, the height and the width of a PRep are fixed to be multiple of 4 and 8, respectively. Considering an architecture with P parameters and having

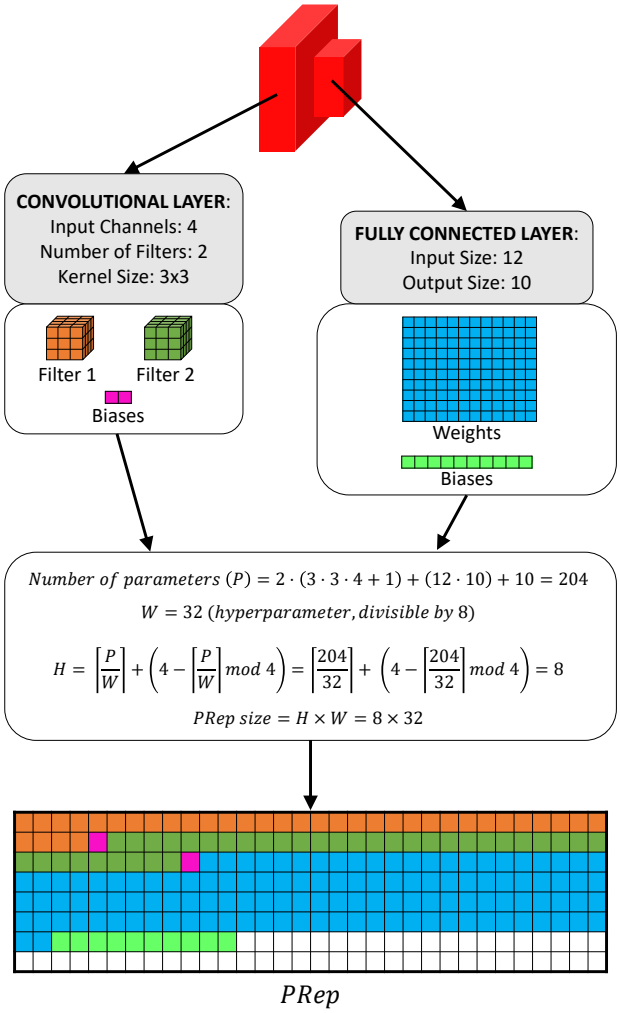


FIGURE C.2: Algorithm to compute the PRep of a given instance. We consider here a toy architecture made out of one convolutional layer and one fully connected layer and we fix the PRep width to 32. White cells represent padding with constant value 0.

fixed the width of the PRep to a number W (divisible by 8), the minimum necessary height of the PRep can be computed as $\lceil \frac{P}{W} \rceil$, adding a padding of $4 - (\lceil \frac{P}{W} \rceil \mod 4)$ rows, if needed, to fulfill the divisibility by 4.

Given an instance of a neural network and a chosen PRep size, the algorithm to produce the instance PRep is straightforward: parameters from all the layers of the instance are copied in the matrix in sequence one row after the other and final zero-padding is added as needed to match the required size.

C.3 Network architectures

Image Classification. Fig. C.3 and C.4 present the architecture of the neural networks used in our experiments dealing with image classification, i.e. LeNetLike [161], VanillaCNN and ResNet8/32/56 [40]. Each convolutional layer is presented in the form CONV (in I, out O, k K, s S, p P), where

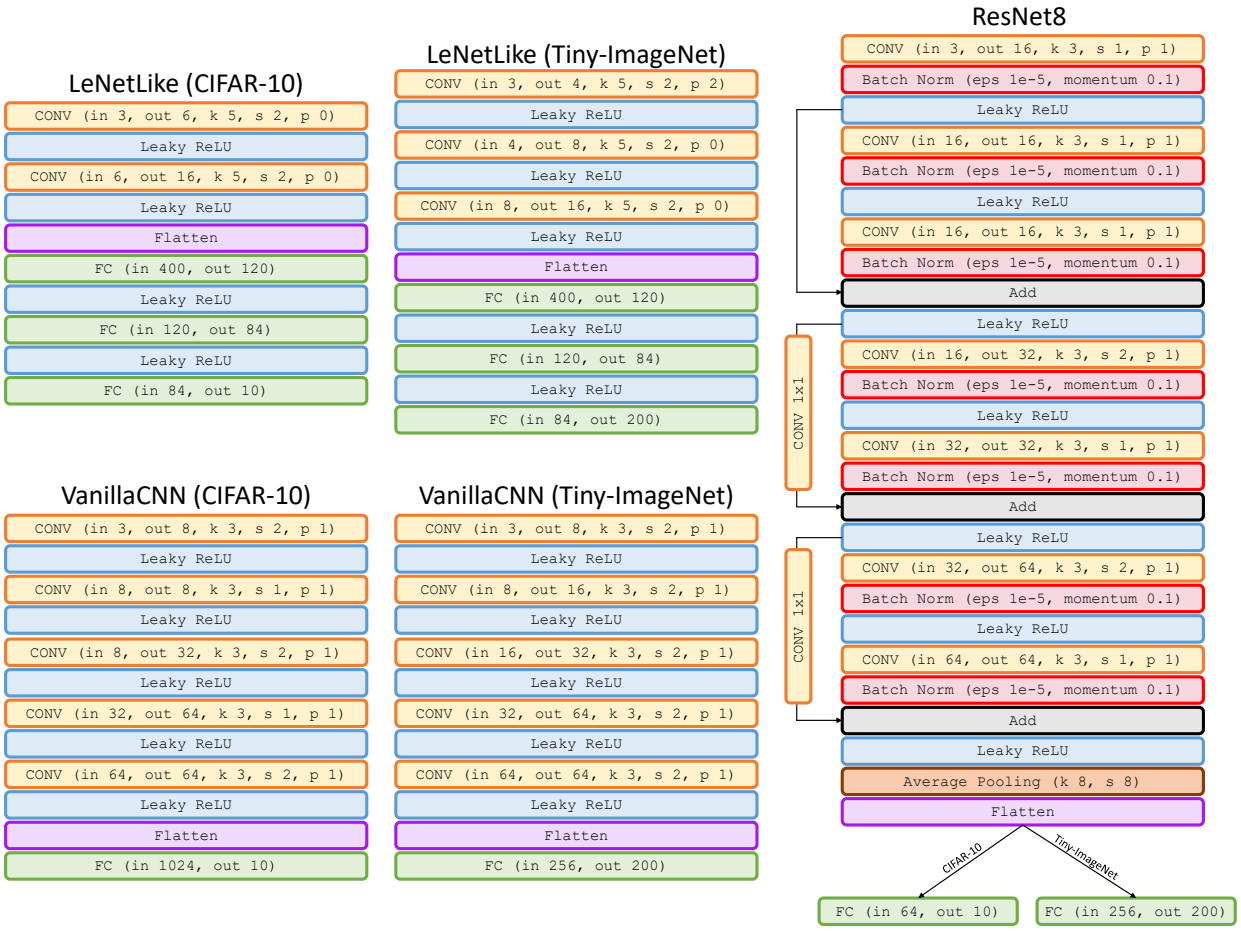


FIGURE C.3: Architectures used in our experiments dealing with image classification. Top left: LeNetLike, bottom left: VanillaCNN, right: ResNet8.

I, O, K, S and P represent input channels, number of filters, kernel size, stride and padding, respectively. Fully Connected layers, instead, are reported in the form FC (in I, out O), where I and O stand for input and output units, respectively . Average pooling is shown as Average Pooling (k K, s S), where K is kernel size and S is stride. Finally, CONV 1×1 represents a convolutional layer with kernel size 1 used to adapt feature maps in residual connections.

3D SDF Regression. As far as 3D SDF regression is concerned, we use simple Multilayer Perceptrons (MLPs) composed of a single hidden layer with 256 nodes. Following [140], we use a periodic activation function between the input layer and the hidden layer and between the hidden layer and the output layer. No activation function is applied instead on the final outputs.

C.4 Image Classification: Experiment Details

Images Datasets. To test our general framework in image classification, we make use of the CIFAR-10 [160] and Tiny-ImageNet [159] datasets. CIFAR-10 is composed of 60K 32×32 colour images, 50K for training and 10K for test, categorized in 10 classes. For our experiments we obtain a validation set by splitting the training set in 40K images for training and 10K for validation, while we keep unchanged the test set. Tiny-ImageNet consists of 100K colour images with resolution 64×64, categorized in 200 different classes. We split the training set in 80K images for

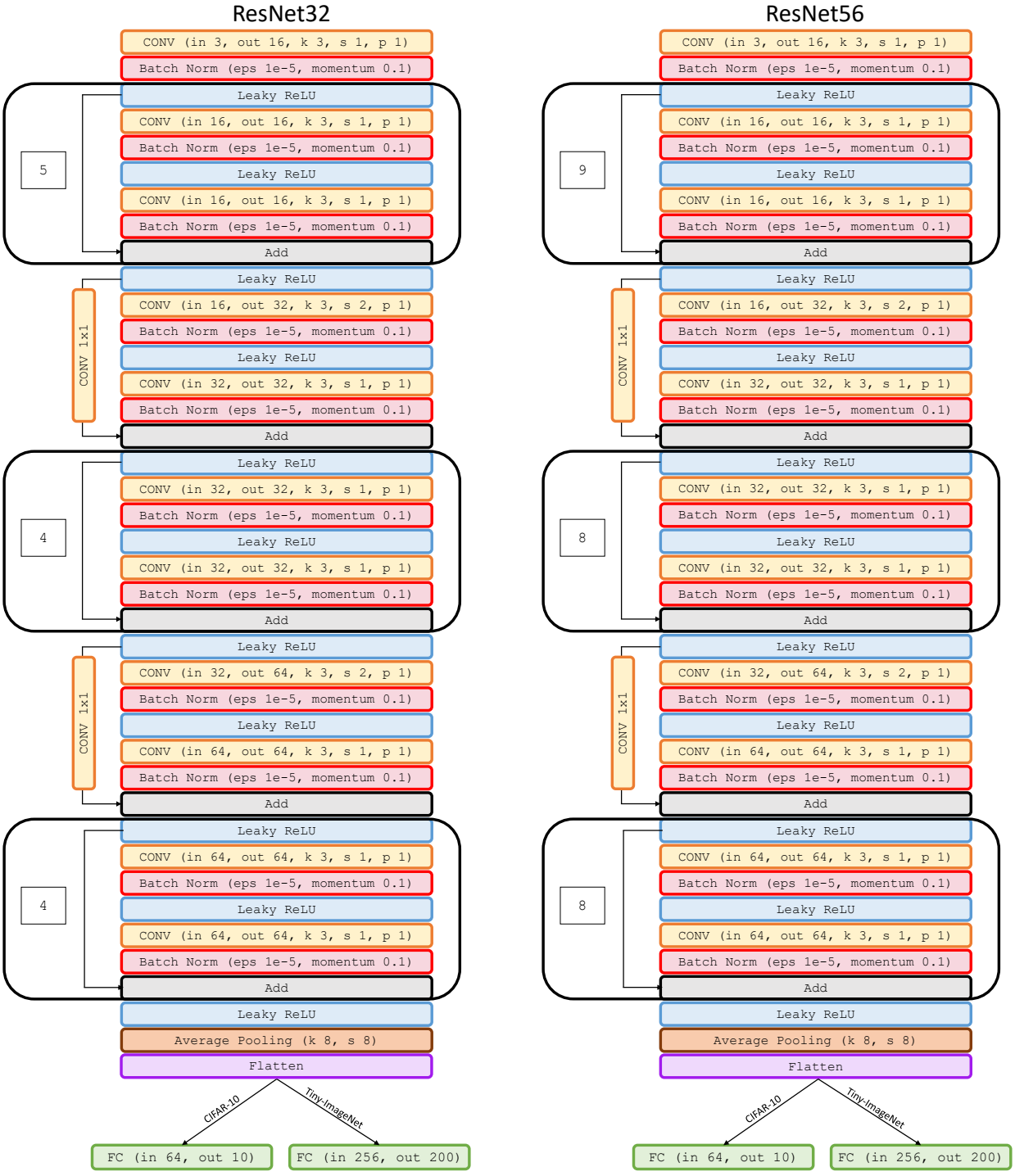


FIGURE C.4: Architectures used in our experiments dealing with image classification. Left: ResNet32, right: ResNet56.

training and 20K images for validation and use the 10K images of the provided validation set for testing. With both datasets, we follow a standard data augmentation regime [163] and use a batch size of 128.

Nets Datasets. In the Single-Architecture setting, we train the instances in input to NetSpace with the Adam optimizer [111] and constant learning rate set to 0.0001 for a number of epochs that vary from 1 to 600 epochs, obtaining instances with weights and performances varying smoothly across the training iterations. Then we use 100 instances for training, 16 for validation and 16 for test. In the Multi-Architecture setting, we aim at embedding only instances with high performances. Accordingly, we found it more effective to train models with the SGD optimizer for 300 epochs and setting momentum and weight decay to 0.9 and 5e-4, respectively. We set the initial learning rate to 0.05 and decay it by 0.1 at epochs 150, 180 and 210. As discussed in Section 5.4, for the Multi-Architecture setting we define a training dataset with 60 LeNetLike instances, 50 VanillaCNN instances, 60 ResNet8 instances and 100 ResNet32 instances, while for the experiment "Sampling of Unseen Architectures" we use a training set composed of 40 LeNetLike instances and 80 ResNet32 instances. In both Multi-Architecture settings, the validation and test sets are composed of 16 instances of the architectures available during training. In Tab. C.1 we report the accuracy achieved by the trained models on the CIFAR-10 and the Tiny-ImageNet test sets, alongside with the number of parameters of each architecture.

Framework Training. To train NetSpace in the Single-Architecture and in the Multi-Architecture settings, we use the Adam optimizer and a learning rate value of 0.0001, we train for around 1K epochs, and then we use the model with the highest performance on the validation set. The temperature term used in $\mathcal{L}_{kd}$ and in $\mathcal{L}^{\gamma}$ is set to 4, while the size of the meta-batch of instances is set to 8 and to 2 in the Single-Architecture and in the Multi-Architecture settings, respectively. The Latent Space Optimization experiments are conducted by training NetSpace with the Adam optimizer and constant learning rate 0.0001, stopping the training when the accuracy achieved by the optimized network doesn't show any additional improvement.

Computational Time and Resources. For our experiments we used several Nvidia RTX 3090 GPUs. Training the networks to populate the datasets requires approximately 600 GPU hours, while training NetSpace with Tiny-ImageNet requires around 84 GPU hours in the Multi-Architecture setting and around 48 GPU hours in the Single-Architecture setting. Training over CIFAR-10, instead, requires less time, with less then 48 and 36 GPU hours respectively in the Multi-Architecture and in the Single-Architecture settings. Finally, the Latent Space Optimization experiments require few GPU hours, but it's possible to observe good improvements over the initial performance already after few minutes of training.

C.5 3D SDF Regression: Experiment Details

3D Shape Dataset. To test NetSpace with networks dealing with 3D SDF regression, we use the ShapeNet dataset [162]. In particular, we use ShapeNetCore, a subset of the full ShapeNet dataset which covers 55 common object categories with about 51,300 unique 3D models. We conduct our experiments on a small subset of ShapeNetCore, consisting of 1000 3D objects from the *chair* category.

CIFAR-10

Net	ClassId	Acc. (Adam)	Acc. (SGD)	# Params
LeNetLike	0	-	70.87 %	62,006
VanillaCNN	1	-	79.77 %	68,818
ResNet8	2	82.79 %	87.13 %	78,042
ResNet32	3	-	92.70 %	466,906
ResNet56	-	-	92.85 %	855,770

Tiny-ImageNet

Net	ClassId	Acc. (Adam)	Acc. (SGD)	# Params
LeNetLike	0	-	22.13 %	79,612
VanillaCNN	1	-	31.32 %	112,856
ResNet8	2	40.51 %	44.38 %	128,792
ResNet32	3	-	54.81 %	517,656
ResNet56	-	-	56.93 %	906,520

TABLE C.1: Models used in our experiments. We show classification accuracies on the CIFAR-10 (top) and Tiny-ImageNet (bottom) test sets alongside the number of parameters for each architecture. For the Single-Architecture setting (Adam optimizer), we report the accuracy achieved by the best performing network between the trained ones, while for the Multi-Architecture setting (SGD optimizer), we report the average accuracy of the networks that compose the training set.

MLP Dataset. The MLP dataset used in our experiments contains 1000 MLP. Each of them is obtained by training a randomly initialized MLP to fit the SDF of a single *chair*, whose ground-truth is computed with the code provided with [141]. The fitting procedure consists in 10,000 gradient descent steps: at each step the MLP is queried on 20,000 random 3D coordinates and is asked to regress the SDF value for each of them. Then, the parameters of the MLP are optimized by Adam [111], using as loss function the mean squared error between the predictions and the ground-truth. The learning rate is initially set to 0.0001 and multiplied by 0.9 every 1000 steps.

Framework Training. NetSpace is trained on the 1000 MLPs with the protocol described in Section 5.3, using Adam [111] with learning rate set to 0.0001. During training, we evaluate the performance of NetSpace by comparing directly the predictions of the output MLPs with those of the input MLPs: by querying input and output MLPs with the same random coordinates, we can compute the percentage of predictions of the output MLPs that are sufficiently close to the values predicted by the input MLPs. We monitor this metric and stop the training when it reaches the value 0.8. Our results are obtained by training for 4000 epochs.

Computational Time and Resources. We adopted Nvidia RTX 3090 GPUs also in the experiments involving SDF regression. The creation of the MLP dataset requires approximatively 20 GPU hours, while training NetSpace requires around 48 GPU hours.

C.6 Fusing batch norm and convolutions

To be able to process architectures including batch norm layers, e.g. ResNet in our experiments, without changing the PRep structure, we decided to fuse batch norm layers with convolutional layers, which is always possible for a trained model since batch norm becomes a frozen affine

transformation at test time. Therefore, when processing ResNet instances in our experiments, we first prepared a dataset of instances trained with batch norm to achieve the best performances and then, by the process described below, we transformed such instances into equivalent ones featuring only plain convolutional layers without batch norm.

If we consider a feature map F with shape $C \times H \times W$, at inference time its batch normalized version $\widehat{F}$ is obtained by computing at each spatial location i, j:

$$\begin{pmatrix} \widehat{F}_{1,i,j} \\ \widehat{F}_{2,i,j} \\ \vdots \\ \widehat{F}_{C,i,j} \end{pmatrix} = W_{BN} \cdot \begin{pmatrix} F_{1,i,j} \\ F_{2,i,j} \\ \vdots \\ F_{C,i,j} \end{pmatrix} + b_{BN} \tag{C.1}$$

with

$$W_{BN} = \begin{pmatrix} \frac{\gamma_1}{\sqrt{\widehat{\sigma}_1^2+\epsilon}} & & & \\ & \frac{\gamma_2}{\sqrt{\widehat{\sigma}_2^2+\epsilon}} & & \\ & & \ddots & \\ & & & \frac{\gamma_C}{\sqrt{\widehat{\sigma}_C^2+\epsilon}} \end{pmatrix} \tag{C.2}$$

$$b_{BN} = \begin{pmatrix} \beta_1 - \gamma_1 \frac{\widehat{\mu}_1}{\sqrt{\widehat{\sigma}_1^2+\epsilon}} \\ \beta_2 - \gamma_2 \frac{\widehat{\mu}_2}{\sqrt{\widehat{\sigma}_2^2+\epsilon}} \\ \vdots \\ \beta_C - \gamma_C \frac{\widehat{\mu}_C}{\sqrt{\widehat{\sigma}_C^2+\epsilon}} \end{pmatrix} \tag{C.3}$$

where $\widehat{\mu}_c$, $\widehat{\sigma}_c^2$, β_c and γ_c ($c = 1, 2, \ldots, C$) are respectively the mean, variance and batch norm parameters computed during training for the channel c of the feature map. From this formulation, we can see that batch norm can be implemented as a 1×1 convolution and therefore, when batch norm comes after another convolution as in ResNet, we can fuse these two convolutions into a single one.

We can express a convolutional layer with kernel size k processing the $C_{prev} \times k \times k$ volume at the spatial location (i, j) of a feature map F_{prev} with C_{prev} channels to produce the feature map $\tilde{F}$ with C output channels as an affine transfomation

$$\tilde{f}_{i,j} = W_{conv} f_{i,j} + b_{conv}, \tag{C.4}$$

where $W_{conv} \in \mathbb{R}^{C \times (C_{prev} k^2)}$, $b_{conv} \in \mathbb{R}^C$ and $f_{i,j}$ represents the area of size $C_{prev} \times k \times k$ around cell (i, j) reshaped as a $(C_{prev} k^2)$-dimensional vector.

If the batch norm defined by $W_{BN} \in \mathbb{R}^{C \times C}$ and $b_{BN} \in \mathbb{R}^C$ presented in Eq. C.2 and C.3 comes after such convolutional layer, the normalized values $\widehat{f}_{i,j} \in \mathbb{R}^C$ at cell (i, j) of its output feature map can be computed as

$$\begin{aligned} \widehat{f}_{i,j} &= W_{BN} \tilde{f}_{i,j} + b_{BN} \\ &= W_{BN} (W_{conv} f_{i,j} + b_{conv}) + b_{BN}. \end{aligned} \tag{C.5}$$

Hence, it is possible to replace every convolutional layer (with weights W_{conv} and b_{conv}) followed by a batch norm layer (whose weights can be shaped in W_{BN} and b_{BN} as described above) with a single convolutional layer whose parameters W and b can be computed as:

$$W = W_{BN}\, W_{conv} \tag{C.6}$$

$$b = W_{BN}\, b_{conv} + b_{BN} \tag{C.7}$$

C.7 Visualizing Networks as Images

As in our framework network instances are represented as PRep tensors, they can be visualized as images. Thus, in this section we highlight some properties of NetSpace by visualizing input and output instances as images. In particular, following the ResNet8 Single-Architecture (Image classification) and ResNet32 Multi-Architecture trainings, we take some instances from the test set and obtain their PReps along with those of the corresponding instances predicted by NetSpace. As explained in Sec. C.2 of this document, such representations are 2D matrices, that we reshaped to obtain images of form factors amenable to clear visualization. Then, as shown in Fig. C.5 and C.6, we represent parameters according to a standard colormap.

From these results we can highlight some interesting properties about our framework. Firstly, we observe that PReps predicted by NetSpace are significantly different w.r.t. the input ones: as we did not use any reconstruction loss in the learning objective, NetSpace learnt to predict instances which behave like the input ones but that are different in terms of parameter values. Secondly, we can see that PReps corresponding to different instances can indeed be visualized as different images, which suggests that, perhaps, in future work these images may be used as proxies for neural networks instances, so that training or fine-tuning or distillation may be realized by learning to generate images.

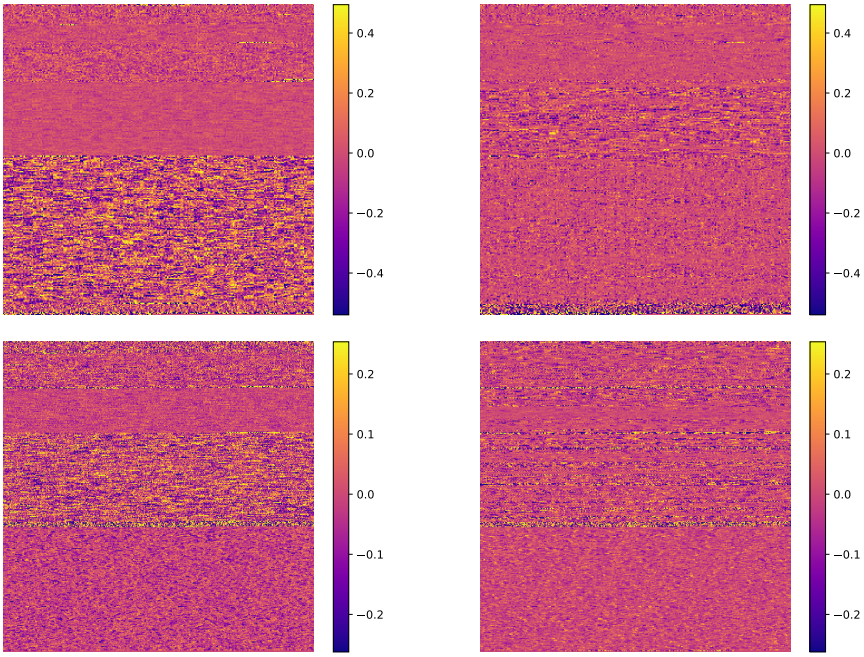


FIGURE C.5: Visualization of the PRep of a ResNet8 instance in the Single-Architecture setting (Image classification): top, CIFAR-10, bottom, Tiny-ImageNet. The target PRep on the left is given in input to NetSpace, that produces the predicted PRep on the right.

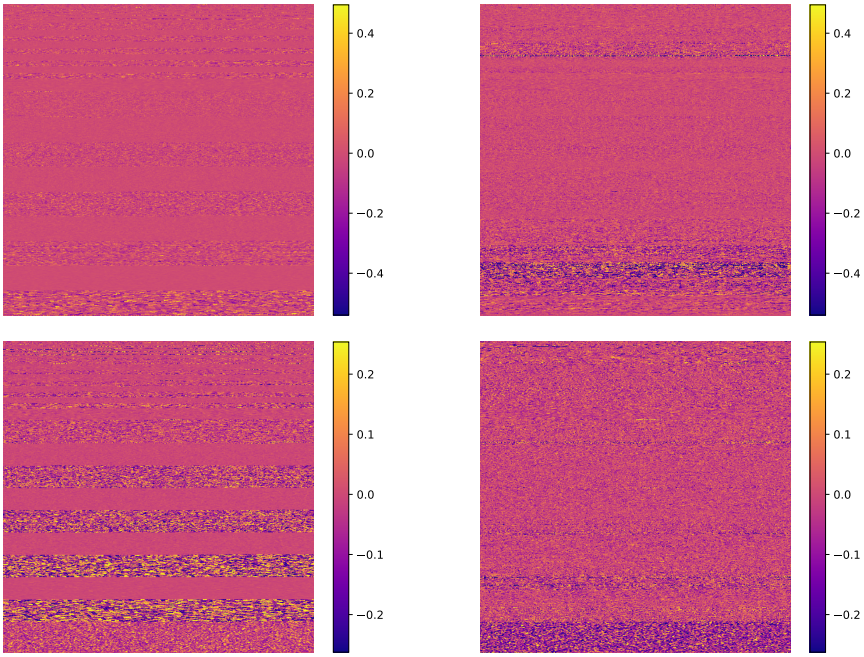


FIGURE C.6: Visualization of the PRep of a ResNet32 instance in the Multi-Architecture setting: top, CIFAR-10, bottom, Tiny-ImageNet. The target PRep on the left is given in input to NetSpace, that produces the predicted PRep on the right.

Bibliography

[1] Ayan Kumar Bhunia, Pinaki Nath Chowdhury, Aneeshan Sain, Yongxin Yang, Tao Xiang, and Yi-Zhe Song. "More photos are all you need: Semi-supervised learning for fine-grained sketch based image retrieval". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 4247–4256.

[2] Peng Xu, Kun Liu, Tao Xiang, Timothy M Hospedales, Zhanyu Ma, Jun Guo, and Yi-Zhe Song. "Fine-Grained Instance-Level Sketch-Based Video Retrieval". In: *arXiv preprint arXiv:2002.09461* (2020).

[3] Xiaoyu Xiang, Ding Liu, Xiao Yang, Yiheng Zhu, Xiaohui Shen, and Jan P Allebach. "Adversarial Open Domain Adaptation for Sketch-to-Photo Synthesis". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2022, pp. 1434–1444.

[4] Sheng-Yu Wang, David Bau, and Jun-Yan Zhu. "Sketch your own gan". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 14050–14060.

[5] Anran Qi, Yulia Gryaditskaya, Jifei Song, Yongxin Yang, Yonggang Qi, Timothy M Hospedales, Tao Xiang, and Yi-Zhe Song. "Toward fine-grained sketch-based 3d shape retrieval". In: *IEEE Transactions on Image Processing* 30 (2021).

[6] Rui Xu, Zongyan Han, Le Hui, Jianjun Qian, and Jin Xie. "Domain Disentangled Generative Adversarial Network for Zero-Shot Sketch-Based 3D Shape Retrieval". In: *arXiv preprint arXiv:2202.11948* (2022).

[7] Song-Hai Zhang, Yuan-Chen Guo, and Qing-Wen Gu. "Sketch2Model: View-aware 3d modeling from single free-hand sketches". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 6012–6021.

[8] Zezhou Cheng, Menglei Chai, Jian Ren, Hsin-Ying Lee, Kyle Olszewski, Zeng Huang, Subhransu Maji, and Sergey Tulyakov. "Cross-modal 3d shape generation and manipulation". In: *European Conference on Computer Vision*. Springer. 2022, pp. 303–321.

[9] Di Kong, Qiang Wang, and Yonggang Qi. "A Diffusion-ReFinement Model for Sketch-to-Point Modeling". In: *Proceedings of the Asian Conference on Computer Vision*. 2022, pp. 1522–1538.

[10] Yue Zhong, Yulia Gryaditskaya, Honggang Zhang, and Yi-Zhe Song. "Deep sketch-based modeling: Tips and tricks". In: *2020 International Conference on 3D Vision (3DV)*. IEEE. 2020, pp. 543–552.

[11] Jiayun Wang, Jierui Lin, Qian Yu, Runtao Liu, Yubei Chen, and Stella X Yu. "3D Shape Reconstruction from Free-Hand Sketches". In: *arXiv preprint arXiv:2006.09694* (2020).

[12] Benoit Guillard, Edoardo Remelli, Pierre Yvernay, and Pascal Fua. "Sketch2Mesh: Reconstructing and Editing 3D Shapes from Sketches". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 13023–13032.

[13] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. "Learning transferable visual models from natural language supervision". In: *International Conference on Machine Learning*. PMLR. 2021.

[14] Jonathan Frankle and Michael Carbin. "The lottery ticket hypothesis: Finding sparse, trainable neural networks". In: *arXiv preprint arXiv:1803.03635* (2018).

[15] Tao Chen, Ming-Ming Cheng, Ping Tan, Ariel Shamir, and Shi-Min Hu. "Sketch2photo: Internet image montage". In: *ACM transactions on graphics (TOG)* 28.5 (2009), pp. 1–10.

[16] Mathias Eitz, James Hays, and Marc Alexa. "How do humans sketch objects?" In: *ACM Transactions on graphics (TOG)* 31.4 (2012), pp. 1–10.

[17] Rui Hu and John Collomosse. "A performance evaluation of gradient field hog descriptor for sketch based image retrieval". In: *Computer Vision and Image Understanding* 117.7 (2013), pp. 790–806.

[18] Bo Li, Afzal Godil, Masaki Aono, X Bai, Takahiko Furuya, L Li, Roberto Javier López-Sastre, Henry Johan, Ryutarou Ohbuchi, Carolina Redondo-Cabrera, et al. "SHREC'12 Track: Generic 3D Shape Retrieval." In: *3DOR@ Eurographics*. 2012, pp. 119–126.

[19] HyoJong Shin and Takeo Igarashi. "Magic canvas: interactive design of a 3-d scene prototype from freehand sketches". In: *Proceedings of Graphics Interface 2007*. 2007, pp. 63–70.

[20] Kun Xu, Kang Chen, Hongbo Fu, Wei-Lun Sun, and Shi-Min Hu. "Sketch2Scene: Sketch-based co-retrieval and co-placement of 3D models". In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), pp. 1–15.

[21] Mohamed R Amer, Siavash Yousefi, Raviv Raich, and Sinisa Todorovic. "Monocular extraction of 2.1 d sketch using constrained convex optimization". In: *International Journal of Computer Vision* 112.1 (2015), pp. 23–42.

[22] Johanna Delanoy, Mathieu Aubry, Phillip Isola, Alexei A Efros, and Adrien Bousseau. "3d sketching using multi-view deep volumetric prediction". In: *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1.1 (2018), pp. 1–22.

[23] Yulia Gryaditskaya, Mark Sypesteyn, Jan Willem Hoftijzer, Sylvia C Pont, Frédo Durand, and Adrien Bousseau. "OpenSketch: a richly-annotated dataset of product design sketches." In: *ACM Trans. Graph.* 38.6 (2019), pp. 232–1.

[24] Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays. "The sketchy database: learning to retrieve badly drawn bunnies". In: *ACM Transactions on Graphics (TOG)* 35.4 (2016), pp. 1–12.

[25] Kunihiko Fukushima. "Neural network model for a mechanism of pattern recognition unaffected by shift in position-Neocognitron". In: *IEICE Technical Report, A* 62.10 (1979), pp. 658–665.

[26] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[27] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[29] Alex Graves. "Generating sequences with recurrent neural networks". In: *arXiv preprint arXiv:1308.0850* (2013).

[30] David Ha and Douglas Eck. "A neural representation of sketch drawings". In: *arXiv preprint arXiv:1704.03477* (2017).

[31] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[32] Michael I Jordan. "Serial order: A parallel distributed processing approach". In: *Advances in psychology*. Vol. 121. Elsevier, 1997, pp. 471–495.

[33] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[34] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. "Generating sentences from a continuous space". In: *arXiv preprint arXiv:1511.06349* (2015).

[35] Alexander Wang, Mengye Ren, and Richard Zemel. "Sketchembednet: Learning novel concepts by imitating drawings". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 10870–10881.

[36] Leo Sampaio Ferraz Ribeiro, Tu Bui, John Collomosse, and Moacir Ponti. "Sketchformer: Transformer-based representation for sketched structure". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 14153–14162.

[37] Ayan Kumar Bhunia, Ayan Das, Umar Riaz Muhammad, Yongxin Yang, Timothy M Hospedales, Tao Xiang, Yulia Gryaditskaya, and Yi-Zhe Song. "Pixelor: A Competitive Sketching AI Agent. So you think you can sketch?" In: *ACM Transactions on Graphics (TOG)* 39.6 (2020), pp. 1–15.

[38] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. "Wasserstein auto-encoders". In: *arXiv preprint arXiv:1711.01558* (2017).

[39] Ayan Kumar Bhunia, Pinaki Nath Chowdhury, Yongxin Yang, Timothy M Hospedales, Tao Xiang, and Yi-Zhe Song. "Vectorization and rasterization: Self-supervised learning for sketch and handwriting". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 5672–5681.

[40] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[41] Peng Xu, Chaitanya K Joshi, and Xavier Bresson. "Multigraph transformer for free-hand sketch recognition". In: *IEEE Transactions on Neural Networks and Learning Systems* (2021).

[42] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. "The graph neural network model". In: *IEEE transactions on neural networks* 20.1 (2008), pp. 61–80.

[43] Thomas N Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907* (2016).

[44] Lumin Yang, Jiajie Zhuang, Hongbo Fu, Xiangzhi Wei, Kun Zhou, and Youyi Zheng. "Sketchgnn: Semantic sketch segmentation with graph neural networks". In: *ACM Transactions on Graphics (TOG)* 40.3 (2021), pp. 1–13.

[45] Yonggang Qi, Guoyao Su, Pinaki Nath Chowdhury, Mingkang Li, and Yi-Zhe Song. "Sketchlattice: Latticed representation for sketch manipulation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 953–961.

[46] Hua Zhang, Si Liu, Changqing Zhang, Wenqi Ren, Rui Wang, and Xiaochun Cao. "Sketchnet: Sketch classification with web images". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 1105–1113.

[47] Gaurav Jain, Shivang Chopra, Suransh Chopra, and Anil Singh Parihar. "Transsketchnet: Attention-based sketch recognition using transformers". In: *ECAI 2020*. IOS Press, 2020, pp. 2907–2908.

[48] Ying Zheng, Hongxun Yao, Xiaoshuai Sun, Shengping Zhang, Sicheng Zhao, and Fatih Porikli. "Sketch-specific data augmentation for freehand sketch recognition". In: *Neurocomputing* 456 (2021), pp. 528–539.

[49] Lan Yang, Kaiyue Pang, Honggang Zhang, and Yi-Zhe Song. "Sketchaa: Abstract representation for abstract sketches". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10097–10106.

[50] Qian Yu, Feng Liu, Yi-Zhe Song, Tao Xiang, Timothy M Hospedales, and Chen-Change Loy. "Sketch me that shoe". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 799–807.

[51] Jungwoo Choi, Heeryon Cho, Jinjoo Song, and Sang Min Yoon. "Sketchhelper: Real-time stroke guidance for freehand sketch retrieval". In: *IEEE Transactions on Multimedia* 21.8 (2019), pp. 2083–2092.

[52] Kaiyue Pang, Yongxin Yang, Timothy M Hospedales, Tao Xiang, and Yi-Zhe Song. "Solving mixed-modal jigsaw puzzle for fine-grained sketch-based image retrieval". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 10347–10355.

[53] Aneeshan Sain, Ayan Kumar Bhunia, Yongxin Yang, Tao Xiang, and Yi-Zhe Song. "Stylemeup: Towards style-agnostic sketch-based image retrieval". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 8504–8513.

[54] John P Collomosse, Graham McNeill, and Yu Qian. "Storyboard sketches for content based video retrieval". In: *2009 IEEE 12th International Conference on Computer Vision*. IEEE. 2009, pp. 245–252.

[55] Rui Hu and John Collomosse. "Motion-sketch based video retrieval using a trellis levenshtein distance". In: *2010 20th International Conference on Pattern Recognition*. IEEE. 2010, pp. 121–124.

[56] Rui Hu, Stuart James, and John Collomosse. "Annotated free-hand sketches for video retrieval using object semantics and motion". In: *International Conference on Multimedia Modeling*. Springer. 2012, pp. 473–484.

[57] Songwei Ge, Vedanuj Goswami, C Lawrence Zitnick, and Devi Parikh. "Creative Sketch Generation". In: *arXiv preprint arXiv:2011.10039* (2020).

[58] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial networks". In: *In Advances in Neural Information Processing Systems (NIPS)* (2014).

[59] Ankan Kumar Bhunia, Salman Khan, Hisham Cholakkal, Rao Muhammad Anwer, Fahad Shahbaz Khan, Jorma Laaksonen, and Michael Felsberg. "Doodleformer: Creative sketch drawing with transformers". In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVII*. Springer. 2022, pp. 338–355.

[60] Wamiq Para, Shariq Bhat, Paul Guerrero, Tom Kelly, Niloy Mitra, Leonidas J Guibas, and Peter Wonka. "Sketchgen: Generating constrained cad sketches". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 5077–5088.

[61] Wengling Chen and James Hays. "Sketchygan: Towards diverse and realistic sketch to image synthesis". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 9416–9425.

[62] Chengying Gao, Qi Liu, Qi Xu, Limin Wang, Jianzhuang Liu, and Changqing Zou. "SketchyCOCO: image generation from freehand scene sketches". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 5174–5183.

[63] Runtao Liu, Qian Yu, and Stella X Yu. "Unsupervised sketch to photo synthesis". In: *European Conference on Computer Vision*. Springer. 2020, pp. 36–52.

[64] Bingchen Liu, Kunpeng Song, Yizhe Zhu, and Ahmed Elgammal. "Sketch-to-art: Synthesizing stylized art images from sketches". In: *Proceedings of the Asian Conference on Computer Vision*. 2020.

[65] Shu-Yu Chen, Wanchao Su, Lin Gao, Shihong Xia, and Hongbo Fu. "DeepFaceDrawing: Deep generation of face images from sketches". In: *ACM Transactions on Graphics (TOG)* 39.4 (2020), pp. 72–1.

[66] Shu-Yu Chen, Feng-Lin Liu, Yu-Kun Lai, Paul L Rosin, Chunpeng Li, Hongbo Fu, and Lin Gao. "Deepfaceediting: Deep face generation and editing with disentangled geometry and appearance control". In: *arXiv preprint arXiv:2105.08935* (2021).

[67] Leo Sampaio Ferraz Ribeiro, Tu Bui, John Collomosse, and Moacir Ponti. "Scene designer: a unified model for scene search and synthesis from sketch". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 2424–2433.

[68] Fang Liu, Changqing Zou, Xiaoming Deng, Ran Zuo, Yu-Kun Lai, Cuixia Ma, Yong-Jin Liu, and Hongan Wang. "SceneSketcher: Fine-Grained Image Retrieval with Scene Sketches". In: *European Conference on Computer Vision*. Springer. 2020, pp. 718–734.

[69] Pinaki Nath Chowdhury, Aneeshan Sain, Ayan Kumar Bhunia, Tao Xiang, Yulia Gryaditskaya, and Yi-Zhe Song. "Fs-coco: Towards understanding of freehand sketches of common objects in context". In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VIII*. Springer. 2022, pp. 253–270.

[70] Mathias Eitz, Ronald Richter, Tamy Boubekeur, Kristian Hildebrand, and Marc Alexa. "Sketch-based shape retrieval". In: *ACM Trans. Graph.* 31.4 (2012).

[71] Sang Min Yoon, Maximilian Scherer, Tobias Schreck, and Arjan Kuijper. "Sketch-based 3D model retrieval using diffusion tensor fields of suggestive contours". In: *Proc. IEEE Trans. Multimedia*. 2010, pp. 193–200.

[72] Dawei Lu, Huadong Ma, and Huiyuan Fu. "Efficient Sketch-based 3D shape retrieval via view selection". In: *Pacific-Rim Conference on Multimedia*. Springer. 2013.

[73] Bo Li, Yijuan Lu, Chunyuan Li, Afzal Godil, Tobias Schreck, Masaki Aono, Martin Burtscher, Hongbo Fu, Takahiko Furuya, Henry Johan, et al. "SHREC'14 track: Extended large scale sketch-based 3D shape retrieval". In: *3D Object Retrie.* Vol. 2014. 2014.

[74] Fang Wang, Le Kang, and Yi Li. "Sketch-based 3d shape retrieval using convolutional neural networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1875–1883.

[75] Long Zhao, Shuang Liang, Jinyuan Jia, and Yichen Wei. "Learning best views of 3D shapes from sketch contour". In: *The Visual Computer* 31.6 (2015).

[76] Lei Li, Zhe Huang, Changqing Zou, Chiew-Lan Tai, Rynson WH Lau, Hao Zhang, Ping Tan, and Hongbo Fu. "Model-driven sketch reconstruction with structure-oriented retrieval". In: *SIGGRAPH ASIA 2016 Technical Briefs*. 2016.

[77] Fan Zhu, Jin Xie, and Yi Fang. "Learning Cross-Domain Neural Networks for Sketch-Based 3D Shape Retrieval". In: *Proc. Associat. Adv. Artif. Intell.* 2016.

[78] Jin Xie, Guoxian Dai, Fan Zhu, and Yi Fang. "Learning barycentric representations of 3D shapes for sketch-based 3d shape retrieval". In: *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.* 2017.

[79] Guoxian Dai, Jin Xie, Fan Zhu, and Yi Fang. "Deep Correlated Metric Learning for Sketch-based 3D Shape Retrieval". In: *Proc. Associat. Adv. Artif. Intell.* 2017.

[80] Xinwei He, Yang Zhou, Zhichao Zhou, Song Bai, and Xiang Bai. "Triplet-Center Loss for Multi-View 3D Object Retrieval". In: *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.* 2018.

[81] A. Qi, Y. Song, and T. Xiang. "Semantic Embedding for Sketch-Based 3D Shape Retrieval". In: *Proc. Brit. Mach. Vis. Conf.* 2018.

[82] Yinjie Lei, Ziqin Zhou, Pingping Zhang, Yulan Guo, Zijun Ma, and Lingqiao Liu. "Deep point-to-subspace metric learning for sketch-based 3D shape retrieval". In: *Pattern Recognition* 96 (2019).

[83] Jiaxin Chen, Jie Qin, Li Liu, Fan Zhu, Fumin Shen, Jin Xie, and Ling Shao. "Deep sketch-shape hashing with segmented 3d stochastic viewing". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019.

[84] Yongzhe Xu, Jiangchuan Hu, Kanoksak Wattanachote, Kun Zeng, and YongYi Gong. "Sketch-based shape retrieval via best view selection and a cross-domain similarity measure". In: *IEEE Transactions on Multimedia* 22.11 (2020).

[85] Zhixiang Chen, Haifeng Zhao, Yan Zhang, Guozi Sun, and Tianjian Wu. "Self-supervised Learning for Sketch-Based 3D Shape Retrieval". In: *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*. Springer. 2022.

[86] Yue Zhao, Qi Liang, Ruixin Ma, Weizhi Nie, and Yuting Su. "JFLN: Joint Feature Learning Network for 2D sketch based 3D shape retrieval". In: *Journal of Visual Communication and Image Representation* (2022).

[87] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. "Teddy: a sketching interface for 3D freeform design". In: *ACM SIGGRAPH 2006 Courses*. 2006, 11–es.

[88] Xiaoguang Han, Chang Gao, and Yizhou Yu. "Deepsketch2face: A deep learning based sketching system for 3d face and caricature modeling". In: *ACM Transactions on graphics (TOG)* 36.4 (2017), pp. 1–12.

[89] Zhaoliang Lun, Matheus Gadelha, Evangelos Kalogerakis, Subhransu Maji, and Rui Wang. "3d shape reconstruction from sketches via multi-view convolutional networks". In: *2017 International Conference on 3D Vision (3DV)*. IEEE. 2017, pp. 67–77.

[90] Marek Dvorožňák, Daniel Sỳkora, Cassidy Curtis, Brian Curless, Olga Sorkine-Hornung, and David Salesin. "Monster mash: a single-view approach to casual 3D modeling and animation". In: *ACM Transactions on Graphics (TOG)* 39.6 (2020), pp. 1–12.

[91] Changjian Li, Hao Pan, Adrien Bousseau, and Niloy J Mitra. "Free2CAD: parsing freehand drawings into CAD commands". In: *ACM Transactions on Graphics (TOG)* 41.4 (2022), pp. 1–16.

[92] Chenjian Gao, Qian Yu, Lu Sheng, Yi-Zhe Song, and Dong Xu. "SketchSampler: Sketch-Based 3D Reconstruction via View-Dependent Depth Sampling". In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part I*. Springer. 2022, pp. 464–479.

[93] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434* (2015).

[94] Tero Karras, Samuli Laine, and Timo Aila. "A style-based generator architecture for generative adversarial networks". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4401–4410.

[95] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. "Analyzing and improving the image quality of stylegan". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 8110–8119.

[96] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. "Alias-free generative adversarial networks". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 852–863.

[97] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. "Image-to-image translation with conditional adversarial networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1125–1134.

[98] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". In: *Computer Vision (ICCV), 2017 IEEE International Conference on*. 2017.

[99] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein generative adversarial networks". In: *International conference on machine learning*. PMLR. 2017, pp. 214–223.

[100] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. "Improved training of wasserstein gans". In: *Advances in neural information processing systems* 30 (2017).

[101] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. "Which training methods for GANs do actually converge?" In: *International conference on machine learning*. PMLR. 2018, pp. 3481–3490.

[102] Andrew Brock, Jeff Donahue, and Karen Simonyan. "Large scale GAN training for high fidelity natural image synthesis". In: *arXiv preprint arXiv:1809.11096* (2018).

[103] Rameen Abdal, Yipeng Qin, and Peter Wonka. "Image2stylegan: How to embed images into the stylegan latent space?" In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 4432–4441.

[104] Rameen Abdal, Yipeng Qin, and Peter Wonka. "Image2stylegan++: How to edit the embedded images?" In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 8296–8305.

[105] Changqing Zou, Qian Yu, Ruofei Du, Haoran Mo, Yi-Zhe Song, Tao Xiang, Chengying Gao, Baoquan Chen, and Hao Zhang. "Sketchyscene: Richly-annotated scene sketches". In: *Proceedings of the european conference on computer vision (ECCV)*. 2018, pp. 421–436.

[106] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. "Coco-stuff: Thing and stuff classes in context". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 1209–1218.

[107] Changqing Zou, Haoran Mo, Chengying Gao, Ruofei Du, and Hongbo Fu. "Language-based colorization of scene sketches". In: *ACM Transactions on Graphics (TOG)* 38.6 (2019), pp. 1–16.

[108] Ranftl René, Lasinger Katrin, Hafner David, Schindler Konrad, and Vladlen Koltun. "Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer." In: *arXiv preprint arXiv:1907.01341* (2019).

[109] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[110] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pp. 8024–8035. URL: `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[111] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *ICLR* (2015).

[112] 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". In: *arXiv preprint arXiv:1706.08500* (2017).

[113] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. "Image quality assessment: from error visibility to structural similarity". In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.

[114] Alex Wong, Safa Cicek, and Stefano Soatto. "Targeted Adversarial Perturbations for Monocular Depth Prediction". In: *Advances in neural information processing systems*. 2020.

[115] Ning Zhang, Lin Zhang, and Zaixi Cheng. "Towards simulating foggy and hazy images and evaluating their authenticity". In: *International Conference on Neural Information Processing*. Springer. 2017, pp. 405–415.

[116] Meng-Li Shih, Shih-Yang Su, Johannes Kopf, and Jia-Bin Huang. "3d photography using context-aware layered depth inpainting". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 8028–8038.

[117] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. "Learning fine-grained image similarity with deep ranking". In: *Proc. of IEEE/CVF CVPR*. 2014.

[118] Ling Luo, Yulia Gryaditskaya, Yongxin Yang, Tao Xiang, and Yi-Zhe Song. "Fine-Grained VR Sketching: Dataset and Insights." In: *2021 International Conference on 3D Vision (3DV)*. IEEE. 2021.

[119] Ling Luo, Yulia Gryaditskaya, Tao Xiang, and Yi-Zhe Song. "Structure-Aware 3D VR Sketch to 3D Shape Retrieval". In: *arXiv preprint arXiv:2209.09043* (2022).

[120] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. "Multi-view convolutional neural networks for 3d shape recognition". In: *Proceedings of the IEEE international conference on computer vision*. 2015.

[121] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. "Self-Attention Generative Adversarial Networks". In: *CoRR* (2018).

[122] Krystian Mikolajczyk and Cordelia Schmid. "Scale & affine invariant interest point detectors". In: *International journal of computer vision* 60.1 (2004).

[123] Filip Radenovic, Giorgos Tolias, and Ondrej Chum. "Deep shape matching". In: *Proceedings of the european conference on computer vision (eccv)*. 2018, pp. 751–767.

[124] Kristofer Schlachter, Benjamin Ahlbrand, Zhu Wang, Valerio Ortenzi, and Ken Perlin. "Zero-Shot Multi-Modal Artist-Controlled Retrieval and Exploration of 3D Object Sets". In: *arXiv preprint arXiv:2209.00682* (2022).

[125] Patsorn Sangkloy, Wittawat Jitkrittum, Diyi Yang, and James Hays. "A Sketch Is Worth a Thousand Words: Image Retrieval with Text and Sketch". In: *European Conference on Computer Vision*. Springer. 2022, pp. 251–267.

[126] Stanislaw Antol, C Lawrence Zitnick, and Devi Parikh. "Zero-shot learning via visual abstraction". In: *European conference on computer vision*. Springer. 2014, pp. 401–416.

[127] David Ha and Douglas Eck. "A Neural Representation of Sketch Drawings". In: *International Conference on Learning Representations*. 2018.

[128] Pinaki Nath Chowdhury, Aneeshan Sain, Yulia Gryaditskaya, Ayan Kumar Bhunia, Tao Xiang, and Yi-Zhe Song. "FS-COCO: Towards understanding of freehand sketches of common objects in context". In: *ECCV*. 2022.

[129] Kihyuk Sohn. "Improved deep metric learning with multi-class n-pair loss objective". In: *Advances in neural information processing systems* 29 (2016).

[130] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. *ShapeNet: An information-rich 3D model repository*. arXiv:1512.03012. 2015. URL: http://arxiv.org/abs/1512.03012.

[131] *Blender Freestyle: https://docs.blender.org/manual/en/latest/render/freestyle/introduction.html*. URL: https://docs.blender.org/manual/en/latest/render/freestyle/introduction.html.

[132] Yael Vinker, Ehsan Pajouheshgar, Jessica Y. Bo, Roman Christian Bachmann, Amit Haim Bermano, Daniel Cohen-Or, Amir Zamir, and Ariel Shamir. "CLIPasso: Semantically-Aware Object Sketching". In: *ACM Trans. Graph.* 41.4 (2022).

[133] Zongze Wu, Dani Lischinski, and Eli Shechtman. "Stylespace analysis: Disentangled controls for stylegan image generation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 12863–12872.

[134] Yuval Alaluf, Omer Tov, Ron Mokady, Rinon Gal, and Amit Bermano. "Hyperstyle: Stylegan inversion with hypernetworks for real image editing". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 18511–18521.

[135] Tengfei Wang, Yong Zhang, Yanbo Fan, Jue Wang, and Qifeng Chen. "High-fidelity gan inversion for image attribute editing". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 11379–11388.

[136] X Zheng, Yang Liu, P Wang, and Xin Tong. "SDF-StyleGAN: Implicit SDF-Based StyleGAN for 3D Shape Generation". In: *Computer Graphics Forum*. Vol. 41. 5. Wiley Online Library. 2022, pp. 52–63.

[137] Yue Jiang, Dantong Ji, Zhizhong Han, and Matthias Zwicker. "Sdfdiff: Differentiable rendering of signed distance fields for 3d shape optimization". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 1251–1261.

[138] William E Lorensen and Harvey E Cline. "Marching cubes: A high resolution 3D surface construction algorithm". In: *ACM siggraph computer graphics* 21.4 (1987), pp. 163–169.

[139] Tejalal Choudhary, Vipul Mishra, Anurag Goswami, and Jagannathan Sarangapani. "A comprehensive survey on model compression and acceleration". In: *Artificial Intelligence Review* (2020), pp. 1–43.

[140] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. "Implicit neural representations with periodic activation functions". In: *Advances in Neural Information Processing Systems* 33 (2020).

[141] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. "Deepsdf: Learning continuous signed distance functions for shape representation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 165–174.

[142] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. "Occupancy networks: Learning 3d reconstruction in function space". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4460–4470.

[143] 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". In: *European Conference on Computer Vision*. Springer. 2020, pp. 405–421.

[144] G. E. Hinton and R. R. Salakhutdinov. "Reducing the Dimensionality of Data with Neural Networks". In: *Science* 313.5786 (2006), pp. 504–507. ISSN: 0036-8075. DOI: 10.1126/science.1127647. eprint: https://science.sciencemag.org/content/313/5786/504.full.pdf. URL: https://science.sciencemag.org/content/313/5786/504.

[145] Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. "Meta-learning with latent embedding optimization". In: *arXiv preprint arXiv:1807.05960* (2018).

[146] Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C Fowlkes, Stefano Soatto, and Pietro Perona. "Task2vec: Task embedding for meta-learning". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 6430–6439.

[147] David Ha, Andrew Dai, and Quoc V Le. "Hypernetworks". In: *arXiv preprint arXiv:1609.09106* (2016).

[148] David Krueger, Chin-Wei Huang, Riashat Islam, Ryan Turner, Alexandre Lacoste, and Aaron Courville. "Bayesian hypernetworks". In: *arXiv preprint arXiv:1710.04759* (2017).

[149] Christos Louizos and Max Welling. "Multiplicative normalizing flows for variational bayesian neural networks". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 2218–2227.

[150] Luca Bertinetto, João F Henriques, Jack Valmadre, Philip Torr, and Andrea Vedaldi. "Learning feed-forward one-shot learners". In: *Advances in neural information processing systems*. 2016, pp. 523–531.

[151] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. "Dynamic filter networks". In: *Advances in Neural Information Processing Systems*. 2016, pp. 667–675.

[152] Jonathan Lorraine and David Duvenaud. "Stochastic hyperparameter optimization through hypernetworks". In: *arXiv preprint arXiv:1802.09419* (2018).

[153] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. "Smash: one-shot model architecture search through hypernetworks". In: *arXiv preprint arXiv:1708.05344* (2017).

[154] Seung Hyun Lee, Dae Ha Kim, and Byung Cheol Song. "Self-supervised Knowledge Distillation Using Singular Value Decomposition". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 335–350.

[155] Oscar Chang and Hod Lipson. "Neural network quine". In: *Artificial Life Conference Proceedings*. MIT Press. 2018, pp. 234–241.

[156] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. "Once-for-all: Train one network and specialize it for efficient deployment". In: *arXiv preprint arXiv:1908.09791* (2019).

[157] Zhichao Lu, Gautam Sreekumar, Erik Goodman, Wolfgang Banzhaf, Kalyanmoy Deb, and Vishnu Naresh Boddeti. "Neural Architecture Transfer". In: *arXiv preprint arXiv:2005.05859* (2020).

[158] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network". In: *arXiv preprint arXiv:1503.02531* (2015).

[159] Ya Le and Xuan Yang. "Tiny imagenet visual recognition challenge". In: *CS 231N* 7 (2015), p. 7.

[160] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images". In: *Tech Report* (2009).

[161] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[162] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. "Shapenet: An information-rich 3d model repository". In: *arXiv preprint arXiv:1512.03012* (2015).

[163] Yonglong Tian, Dilip Krishnan, and Phillip Isola. "Contrastive representation distillation". In: *arXiv preprint arXiv:1910.10699* (2019).