LEARNING TO GENERATE AND DIFFERENTIATE 3D OBJECTS USING GEOMETRY & LANGUAGE

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Panagiotis (Panos) Achlioptas
August 2021

This dissertation is online at: https://purl.stanford.edu/sr155wq1248

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Leonidas Guibas, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Stefano Ermon

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Silvio Savarese

Approved for the Stanford University Committee on Graduate Studies.

Stacey F. Bent, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Abstract

The physical world surrounding us is extremely complex, with a myriad of unexplained phenomena that seem at times mysterious or even magical. In our quest to understand, analyze and in the end, improve our interactions with our surroundings, we decompose this complex world into tangible entities we call **objects**. From Plato's ancient *Theory of Forms* to the modern rules of *Object-Oriented Programming*, objects with their associated classes and abstractions, have been a pillar of analysis and philosophy. At the same time, human intelligence flourishes and demonstrates much of its elegance in another human construct: that of natural **languages**. Humans have developed their languages to enable them to efficiently communicate with each other for almost anything conceivable: from never-seen imaginative scenarios to pragmatic nuisances regarding their surrounding objects.

My vision and motivation behind this thesis lie in bridging (a modest bit) the gap between these two constructs, language and object entities, in modern-day computers via *learning* algorithms. In this way, this thesis aims at contributing a step forward in the advancement of *Artificial Intelligence* by introducing to the research community, smarter, latent, and oftentimes multi-modal representations of 3D objects, that enhance their capacity to reason about them, with (or without) the aid of language.

Specifically, this thesis aims at introducing new methods and new problems at the intersection of the computer science sub-fields of 3D Vision and computational Linguistics. It starts and dedicates about half of its contents by establishing several novel (deep) Generative Neural Networks that can generate/reconstruct/represent common three-dimensional objects (e.g., a 3D point cloud of chair). These networks give rise to object representations that can improve some of the machines' objects-oriented analytical capacities: e.g., to better classify the objects of a collection, or generate novel object instances, by combining a priori known object-parts, or by meaningful "latent" interpolations among specified objects. The second half of the thesis, taps on these object representations to introduce new problems and machine learning-based solutions for *discriminative object-centric* language-comprehension ("listening"), and language-production ("speaking"). In this way, the

second half complements and extends the first part of the thesis, by exploring *multi-modal*, language-aware, object representations that enable a machine to listen or speak about *object properties* similar to humans.

In summary, **the three most salient contributions of this thesis are the following**. *First*, it introduces the first Generative Adversarial Network concerning the *shape* of everyday objects captured via 3D point clouds and appropriate (and widely adopted) evaluation metrics. *Second*, it introduces the problem and deep-learning-based solutions, for comprehending or generating linguistic references concerning the *shape* of common objects, in *contrastive* contexts i.e., talk about how a chair is different from two similar ones. *Last*, it explores a less controlled and harder scenario of object-based reference in the wild. Namely, it introduces the problem and methods for language comprehension concerning properties of real-world objects residing inside *real-world* 3D scenes, e.g., it builds machines that can understand language concerning, say, the texture of an object or its spatial arrangement. During the journey it took to establish these contributions, we published and explored some highly relevant ideas, parts of which will be used to make a more complete exposition. In short, these papers concern two high-level concepts. First, the creation of "latent spaces" that are aware of the *part-based* structure of 3D objects, e.g., the legs vs. the back of a chair. Second, the creation of latent spaces that exploit known correspondences among objects of a collection, e.g., dense pointwise mappings, which can enhance the latent representation capacity in capturing geometric- shape-differences among objects. As we show with the primary works presented in this thesis, object-centric referential language contains a significant amount of part-based and fine-grained shape understanding – naturally calling for a conceptually deep object learning and justifying the ongoing need for the development of many types of Generative Networks to capture it fully.

Dedication

To my father, ***who always had been, and always will be with me.***

Acknowledgments

This thesis is the aggregate result of the imagination, desire, and work of many people. Some of them are the closest people to my heart; some will remain unknown, and with some, I connected "invisibly" via their books, poems, and works. Finally, with a few, I built a special bond within the Spirit of Science & Art. I want to thank all of them from the bottom of my heart, including those with whom I had disagreements and conflicts. The *Essence* never lies in the final destination. It is the *Journey* that matters, as the poet Cavafy elegantly explains in his poem, *Ithaka*.

Being aware that I will not provide a complete list (how could I?), I will mention here some people that I will never forget, and who played a pivotal role in this *Journey*.

Despoina, without your constant love, I would not have nourished my best instincts that helped me leave behind the beautiful island of Crete. Andreas, Karolos, and Panayotes, you were the real deal guys. Faradia, Yu, Kore, thank you for opening and sharing entire new worlds, so beautiful and different from mine. Katerina Ioakeimidi, George & Thaleia, Marilena Stavrides, Athina Loumou, Magda & Alkis, thank you for helping me transition into living in the US smoothly and excitingly – we share a common background and trajectory, and you had gone through this transition well before me.

Robert West, we had many fun moments at Stanford in my early years. Thank you. Jure Leskovec, thank you too. Your way of approaching research and running a lab like a friend is remarkable. Gill Bejerano, you have a huge heart. Thank you for pushing me to find research areas that I would be passionate about and not settle for less. Harendra Guturu, thank you for your patience and the laughs we shared. I won't forget either of them.

Lin Shao, your motto **to never give up** will always ring in my ears. You helped me find strength in some of my hardest times. Also, when Deep Learning was still a mystery, our lengthy talks were essential for me to dig deeper into it.

Vignesh Ganapathi, you are a (still unattainable!) role model for me. Your genuine way of caring so gently about others is among the most excellent human qualities I encountered at Stanford.

I would also like to thank all my lab mates, and especially those who found time, care (and bravery) to be kind, open, and giving. Olga Diamanti, Fei Xia, Or Litany, Eric Yi, Kaichun Mo, Tolga Birdal, Fan Wang, Justin Solomon, Anastasia Dubrivona, Charles Qi, Davis Rempe, Tanya Glozman, Soren Pirk, Raif Rustamov, Qixing Huang. Fan, Raif, and Qixing, thank you for passing your knowledge on a one-on-one basis when I knew next to nothing about Computational Geometry, Functional Maps, and the like. Olga, Fei, Tolga, and Or, thank you for opening your hearts and sharing your talents and life experiences, which helped me tremendously navigate many situations and have more fun in the process of earning my Ph.D.

To my more (academically) senior friends who really helped me find the silver linings and the delicate balance between doing meaningful research and managing the human factor behind collaborative endeavors:

Ioannis Mitliagkas, your genuine interest in my early findings on the AE "stuff" was absolutely instrumental in making me believe that I can finally do some interesting research. Our work is now cited 560 times, and in its early days, the chances of it never going anywhere – if you hadn't pushed us – was substantial. Thank you, my friend.

Maks Ovsjanikov, thank you for taking a deep interest in shaping me as a researcher. I think our shared passion for excellence and detail-oriented analysis has made our bond always fun and (hopefully, mutually) mentally enriching. I will never forget your personal care in helping me.

Noah Goodman, thank you for all the beautiful things you taught me. The concept of creating pragmatic neural models is among the most elegant scientific matters I have worked in, and you single-handedly opened that door. I have loved all our science-related discussions.

Sam Ocko, you are the brightest combination of a strong mind and strong heart human I met at Stanford. I am honored to be your friend. Our endless walks and discussions on contrastive-oriented learning were the best shared-mind-explorations.

Kristen Grauman, thank you for the enriching experience of hosting me as an intern at FAIR. It was a très-cool Summer. Mike Haley, thank you for the fantastic internship at AutoDesk and for your friendship. It means a lot to me.

I also want to sincerely thank Jay Subramanian, Carrie Petersen, Marianne Siroker, Helen Buendicho, Jam Kiattinant, Andrej Krevl, and all the staff working at CS department administration. You do the impossible every day: supporting the student-faculty interactions in all fundamental ways that enable the department to exist. In the same vein, I want to thank all the personnel who make the SGF-fellowships a possibility (thank you for giving me one!) and Tatiana Hadjiemmanuel from Fulbright Greece.

I want to thank all my (not mentioned above) co-authors, obviously! Mirah Shalah, Raphaël Groscot, Judy Fan, Robert Hawkins, Ahmed Abdelreheem, Mohamed Elhoseiny, Sherif Abdelkarim, Ruqi Huang, Marie-Julie Rakotosaona, Minhyuk Sung, Niloy Mitra, Kilichbek Haydarov, Lia Coleman, Zhenyu Jiang, and Kenneth Church. This thesis is very much *our* research. Ahmed, you are a true friend, with your help, and that of Iro Armeni, Claudia D'Arpino and Dimitri Logothetis the pandemic times were as mild as they could have been. Thank you! Iro Armeni thank you for helping me in so many ways this final year. Ruqi and Maria-Julie, thank you again.

I also want to express my gratitude to the wonderful Turkers of Amazon Mechanical Turk, whose help in curating several of our introduced datasets was of paramount importance.

I also want to thank several fellow researchers and mentors, Angel Chang, Nikos Gkanatsios, Matthias Niessner, Dave Z. Zhen, Manu Danhert, Ioannis Emiris, Alexandros Dimakis, Maria Papadopouli, Manolis Katevenis, Chentai Kao, Ian Huang, Ersin Yumer, Volodymyr Kuleshov, Winnie Lin, Theofanis Karaletsos, Ishan Gupta, George Neofotistos, John Panageas, Antonia Saravanou, Karsten Borgwardt, Bernhard Schölkopf. Angel thank you for always being approachable. Nikos, thank you for all the "meraki".

Furthermore, I want to thank all the people with whom I connected at Stanford, became friends, and with whom I made so many lovely memories: Vasilis Verios, Josh Gevirtz, Emily Graber, Paris Siminelakis, Dimitris Doukas, Manolis Papadakis, Michael Savvas, Maria Roumpani, Nikos Sarilakis, Yannis Yiakoumis, Daniel Speckhard, Kamran Naim, Chris Paskov, Ben Poole, Dimitris Skourtis, Veruschia Mahomed, Dimitris Loufas, Elena Jordan, J. T. Chipman, Marc the Smoker, Danny Goodman, Amir Sadeghian.

It goes without saying that I owe a lot of this result to my besties from my most formative days: Orestes Poulakis, Tasos Kapes, Vasilis Filios, Themos Georgakas; and to my family circle. Especially Iro & Stelios, the two John Achlioptas', Aristotelis and Anafi, Stelios and Maria Achlioptas, Eleni Dimitropoulou, John Dimitriou, the cool Kikitsa, Irene Katsani, Con and all the Manos, Rita & Antonis, George Mazarakos, Pakis, and Haroula. And, of course, my mother, sister, and brother, and the love of my life, Anastasia. For those I won't say much, they know the *Journey* inside out.

I want to also thank my oral committee members. It was a true honor receiving the green light from such great minds. Thank you Silvio Savarese, James Gross, Stefano Ermon, Dan Jurafsky, and Ioannis Mitliagkas.

Lastly, but most importantly, I owe the biggest thank you to my academic advisor Leonidas Guibas. Leo, I feel an immense debt of gratitude towards you, which cannot be put into words. Thank you from the depths of my heart.

August 24th, 2021
Panagiotis Achlioptas

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Object-centric understanding of the world We live in a three-dimensional world. Our ability to interact, analyze, and model our 3D surroundings is remarkable, and despite very significant efforts we are still a long way from making machines that can act as autonomous agents: that can navigate, interact, grasp, and ultimately reason about their surroundings with the ease, robustness, and breadth that humans do. Perhaps this fact should not come as a big surprise considering that humans had a "few" millennia of an evolutionary advantage to hone and improve such skills. Over these millennia, a key concept that emerged in our consciousness (presumably) to facilitate our daily thinking and livelihood was that of **objects**. Humans, effortlessly can break down the enormous complexity of their surroundings by *decomposing* them into stable smaller entities we call objects. Objects can be moved around, arranged, and re-arranged, while preserving their identity, essential form, and function – e.g., a laptop remains a laptop irrespective of where it is placed, or of what color it has. Crucially we compartmentalize objects into classes and categories which are extremely useful abstractions. First, object classes provide a common naming mechanism for their instantiations (objects) which can vary wildly. Second, they ease our understanding of object-properties/attributes, that oftentimes are shared across instances of different classes, e.g., both a spoon and a bird can be green. Third, categories promote a modular treatment of objects by enabling a *relational* understanding among them, e.g, building hierarchies that reflect specialization: a bird is an animal - or, part-whole relations e.g., the legs of a bird or of a chair.

As important this semantic unification of objects under a given category is, or the sharing of common properties and parts across different categories, – mechanisms that focus on the *similarities*

among objects – it is equally interesting to pay close attention to the complementary notion of *"object differences"*. Capturing and characterizing all possible variations across the instances of a **single** class of objects is a conceptually appealing quest that **has driven much of the work presented in this thesis**. After all, how can we attain a complete machine-learnable understanding of objects (even simple inanimate objects like chairs), if we cannot analyze all the ways that chairs are different from each other? Surely, a complete *generative* mechanism/network/paradigm that can produce all imaginable chairs in finite time has to be able to also understand at some level of abstraction how (if) its output products are different. But before we embark in making such a complete generative machine, it is worth asking how exactly, we, humans, **perceive** (as input) and **communicate** (as output) differences among any predefined set of objects?

The above lines delineate the **basic motivation for my thesis**: to make generative and discriminative networks that better understand 3D objects; which in turn is a stepping stone for enriching computers and autonomous agents with human-level *perception*. To this end we first had to actually develop the first generative network operating with 3D point clouds, introduce novel generative networks that exploit the part-based composition of 3D objects, or exploit correspondence-based geometric differences among objects, and then, finally, create data and methods that explore how 3D objects are different in the eyes of a human who is expressing their differences in written language. In the coming paragraphs, I will delve deeper in the intricacies involved behind these choices that constitute this thesis.

1.1.1 Human perception and AI

3D visual perception Human perception implements several cognitive functions that put together attempt to: organize, identify, and interpret all sensory input information we receive from our environment – with the goal to represent and finally understand one's received information. Visual perception in particular, restricts this sensory input information to that coming from vision, which is perhaps the most important of our five senses, considering that most information ($\sim 75\%$) received daily by humans comes from visual signals [43]. Depending on the specific application, modern AI systems usually require several components approximating functions of human intelligence to operate based on sensory mechanisms: perception, prediction, planning, and control. In this thesis we focus on **3D** *visual perception*, where the sensory input comes from various 3D sensing mechanisms such as depth sensors, multi-view stereo, laser range scanners, etc. While 2D (RGB) images are rich in color/texture and are in abundance nowadays - they are also easily affected by illumination artifacts and occlusions resulting from the shape/arrangements of the portrayed objects. After all, 2D images

are only a single slice of our 3D world and like humans who perceived depth, it is natural to attempt to teach an object-centric view-point to computers *directly* in 3D. There are in fact several reasons why a 3D-based treatment of objects and representation-learning is chosen in this thesis. First, objects in 3D sensory inputs preserve (almost perfectly) their intrinsic structure (e.g., part-hierarchies, or object-scene arrangements), and regularities that if exploited correctly can provide an extra cue to 3D visual perception with the potential of improving an intelligent system's operational performance. Secondly, objects acquired with 3D sensory devices maintain their physical scales, and they are less affected by lighting, occlusion, and projection artifacts compared to images; making tasks like object recognition relatively easier compared to 2D-based alternatives. Last but not least, objects encoded in 3D formats like CAD models, provide a "clean" setup to analyze shape properties that are invariant to the actual placement of the objects inside the hosting/ambient 3D space. These properties include quantitities like the shape's curvature or shape-part composition, which in CAD-like models do not become obfuscated by the clutter that typical image data have. Of course, if we ever want to build intelligent agents that co-operate "live" with us, like robots or agents in AR or VR, understanding our *shared* world which is 3D becomes a necessity.

1.1.2 Generative models and perception

Are generative models important for modeling perception? Generative models can in theory enable the modeling of the entire underlying data distribution of a class of objects, or more broadly items, from a limited sample of observations. A good generative model trained with sufficient data can discover and encode the *principles* behind the creation of the data instances (e.g., the rules used by a template-model that assembles chair-parts resulting in a family of chairs), allowing not only to **interpolate** among any two given data samples: filling the "gaps" of the manifold – but to certain degree also go beyond the "seen" samples, and allow **extrapolation**.

A generative model that has managed to acquire even approximately such understanding of the items that is modelling can be an indispensable tool for any machine that attempts to analyze the world. First, it can be used to improve discriminative decision making and stimulus recognition/classification. The are many studies showing improved classification performance with classifiers that exploit a generative network. For instance, for common 3D objects (chairs, tables, cars, etc.), the study of *Wu et al.* [290], and **our own** [6, 5], show that by means of *transfer learning*, it is beneficial to **pre-train** a generative network with self-supervision i.e., without any labels, to create semantically-rich object representations. These representations can be fine-tuned with low-complexity discriminators like a linear SVM, or a single layer Perceptron [234], to achieve significantly better generalization in object

recognition tasks compared to a non-pre-trained approach. This idea of using unsupervised generative modeling to create representations that are useful in downstream tasks, or in general, is not new. Perhaps the first network which actually managed to become deep did so by gradually pre-training its intermediate layers before expanding them with more layers [105] (overcoming this way the optimization problem of exploding/vanishing gradients occurring in deep-nets trained with simple Stochastic Gradient Descent and not carefully chosen weight-initialization [84]). More recently a variety of other complex down-stream tasks ranging from creating object part-segmentations [55] to object localization in real-world scenes [299, 107] appear to benefit from similar distribution **priors** that training with generative paradigms can bring in the optimization landscape (e.g., [147]) of a discriminating task.

It is worth noting that the above approaches typically split the training into separate loss-functions and corresponding training stages. However, there are more and more advocates suggesting that training generative and discriminative models *jointly* [141, 320, 89] can be also effective. For instance, the work of Gordon and Hernández-Lobato [89], develops a framework that allows semi-supervised models (see [47] for a full treatment) to learn from labelled data (discriminatively) and unlabelled data (generatively) simultaneously. A key benefit of their approach is that it can naturally account for the *uncertainty* in the model's predictive distribution, i.e., their discriminative model can provide better **confidence intervals** for its predictions by tapping in its generative counterpart.

This last point is subtle and it relates to another emerging topic where building good generative models can be useful: **Explainable AI** (XAI). In simple words, XAI is artificial intelligence (AI) in which the results of the provided solutions can be understood by humans. It contrasts with the concept of the "black box" in machine learning where oftentimes even its designers cannot explain why an AI arrived at a specific decision [112]. Obviously, models that can reduce their uncertainty on their predictions (as above in [89]) are safer to use and give better hints as for an *interpretation*. In a similar vein another application of generative modeling that aids human-interpretation is that of data visualization and exploration e.g., by using old-school manifold learning algorithms (e.g., PCA, LLE, Isomap) and tools like T-SNE *inside* the underlying latent space of a generative model, one can discover structure in the data (See Figure 3.8 for an example of this coming from **our work** [75]).

Finally, two last broad use cases where generative models can help machines to better model human perception include their application in the problems of: a) content **modification** and b) emulating **creativity/imagination** to do de novo content-creation, or Zero Shot Learning (ZSL) [296]. With the recent developments in image-based GANs, which drastically improved the quality of their output generations [40, 124, 125, 59] we have seen a intense growth in works that utilize GANs and

latent-spaces created with other generative models like Variational-Autoencoders [131], to *modify* content that is provided as input. For instance, such works range from image-face modifications e.g., make a face look happier, younger, etc. [145, 223, 277], to generating novel shoe designs [71], or predicting the melody of a song from its lyrics [314]. On the 3D/objects front the examples from **our works** in Figure 2.3 and Figure 3.5 show how one can use semantic part labels with either of the generative networks we *introduced* for instance to "add" armrests to a given chair. Finally, on the creativity/imagination front, generative networks are starting to find new (very) creative usages like in the works of CompoNet [242] or CGSP [95]. Here the introduced generative networks exploit the factorization of common shapes into parts which are often shared across categories, to generalize and "imagine" reconstructions (CompoNet) or 3D predictions from single images (CGSP) that go beyond the "seen" manifold similar to a ZSL-setup. As we will see in the coming Chapters (5 & 6) when **we** combined language-based signals with (object-centric) generative models the applications of ZSL extend even more due to the inherent compositionality of human languages e.g., see how our neural-listener generalizes robustly on novel categories of objects in Figure 5.7.

1.1.3 Focusing on object differences

Why modeling object differences in terms of shape, location, etc. ? Traditionally the learning literature explores an object-centric understanding of our world by primarily focusing on the *commonalities* among different objects [32, 85]; This is the case in clustering/compression-like schemes such as AutoEncoders, but also to a lesser extent in deep-learning-based classifiers that aim at finding non-linear decision boundaries that exploit the commonalities of same-class objects and embed them in a space that keeps them in geometric proximity (while separating them from other classes) [121, 184]. The situation is more subtle and the focus starts shifting more clearly towards fine-grained *differences* among objects in studies that involve fine-grained attribute classification [309, 310, 298, 305, 205]. In such studies typically the objects of a **single** class are being considered and the emphasis is placed in learning fine-grained properties that characterize different sub-populations of the same class e.g., is a shoe a sandal or a slipper [309]?, or is a human smiling face or not [162]? Of course, it is easy to see that there is a hidden recursion here. Single level classification problems/datasets like the famous 1,000 class problem posed by ImageNet's ILSVRC2012 challenge [68] considers the macro characteristics of the 1,000 involved categories. Once we consider however only the items of a single (high-level) class, e.g., shoes, we can repeat the classification problem at a finer level e.g., sandals vs. slippers, and then once again, as in leather-sandals vs. plastic-sandals, and so on. The limit of this approach ends with each *individual* item of a collection being considered an (atomic) class, giving

rise to the *"Instance-Level Classification Problem"* (ILC). It is very interesting, how in the recent years the ILC has been utilized at the core of most self-supervised (contrastive) algorithms that have attained state-of-the-art results [294, 273, 269].

Contrastive context matters As the above discussion suggests when we are analyzing an object the implicit or explicit objects we are comparing it with matter. Typical fine-grained classification problems rely on an explicit manually curated catalogue of *predefined* labels that are extracted and learned for every object of a given class. ILS drops the requirement of having a predefined label set, but still considers implicitly the entire set of objects as the comparators. An interesting line of works, starting perhaps with the seminal paper of "Relative Attributes" [205], uses a small set of predefined attributes and compares *fixed tuples* of items in a relative manner: e.g, *is 'this' shoe more sporty than 'that'*? To **explicitly** define the contrastive-context is important as it can be used to provide a linear ordering w.r.t. the underlying attribute ("sporty"). A generalization of this idea that further lifts the necessity of having a predefined attribute-set is to use **referential language** to express the signal (label) that distinguishes an objects in a given context. Natural referential language offers an vast amount of ways to refer to an item in discriminating and reasonable manner (understood by another human), giving rice to an **open-ended** label set. Two of **our main works** follow this paradigm (Chapter 5, and Chapter 6). In a nutshell, in these works a human observer inspects a specific indicated item (the "target" object) and contrasts it within a **fixed** set of other items (the "distractors"). The observer is then asked to refer to the target in a way that makes it identifiable by another human who is exposed to the same objects ("stimuli"). By carefully controlling the communication/contrasting context we can induce comparisons and properties regarding the target at different granularities, e.g., when we contrast an office chair with two (similar) office chairs, we can extract nuanced visual differences such as the target's leg relative thickness; but when we contrast the same item with say two arm-chairs we can capture, say, if the target shares similar high-level structural components like arms (e.g., see Figure C.6). Note, that to build in a scalable manner contrasting contexts of different granularities for our work in Chapter 5, we relied on semantic-similarity-based metrics among the compared objects, which were available from **our earlier** work on *language-free* shape-based AutoEncoders (Chapter 2, Section 2.6.1).

Lastly, it is also worth noting that by controlling the context not only one can induce different granularities over a single quality, like properties concerning an object's shape – but can ultimately impose comparisons involving different properties. An example of this is explored in **our** "natural followup" work presented in Chapter 6. There, we ask an observer to write in language how a 3D

object is identifiable among other objects of the same fine-grained-class e.g., an office-chair vs. other office-chairs, but in the context of a **real-world scene**. Such a setup challenges the user to describe properties such as the unique *spatial relation* of the object w.r.t. any object of the scene, or its unique *texture, shape, size*, ..., or *any* other property that makes it identifiable by another human (see Figure 6.6 of Chapter 6 for interesting cases of outlier-looking referential language).

1.1.4 Synergies and obstacles in relating 3D vision and language

Interplay between 3D vision and language The above discussion gives a high-level set of ideas of how one can explicitly define the contrastive context between a set of 3D (visual) objects, and how to use referential language as the means for extracting a differentiating signal. More broadly, relating language and vision in a **joint fashion** has been among the main research foci of multi-modal learning, with applications ranging from instruction-specifications for robots [267, 200, 140], to image-based captioning [123, 193, 169, 312], and language-based content modification [81, 27]. One fundamental difference between vision-based signals and linguistic descriptions of visual content, concerns their **specificity**. Vision signals, modulo acquisition noise, tend to be precise and specific, while common linguistic expressions describing them are almost always significantly underspecified. For instance, one can generate practically an infinite amount of images portraying "a house", or "a small house by a river". In applications where the goal is to associate language with vision, like in image-captioning and related systems [123, 8], it is important to control this discrepancy and attain the desideratum of generating language (or images) that reflect maximally the corresponding visual (or linguistic) details. A neat technique for achieving this goal, was introduced independently and in slightly different contexts in the works of *Vedantam et al.* [276] and *Andreas et al.* [15]. Their key idea is simple and provides an elegant way **for combining a generative network with a discriminative network**: first train a *generative language model* ("neural speaker") by using standard losses and procedures (e.g., Teacher-Forcing [288]). Then, train a separate bi-modal classifier ("neural listener") that can provide a likelihood for any given sentence as being related (describing-well) any given image. Finally, merge the two systems: sample many linguistic expressions from the neural-speaker but select/output the one that has a high-likelihood according to the neural-listener. This approach, which improves the **pragmatic** association between a visual stimulus and its caption/description, was separately applied in **our work** concerning the production of discriminative language for 3D shapes; significantly improving the quality of the produced language (see Figure 5.6). Interestingly, in a very recent and wildly popular work concerning the reverse problem of image-generation from text: DALL-E [227]; per the authors statement the act of ranking the sampled images by an separately

trained language-image discriminator (CLIP [224]) was absolutely *critical* for generating relevant images. It appears thus that combining generative and discriminative networks is an emerging fruitful practice, and ***I hypothesize*** *that in the future more, very significant, findings will stem from it*.

Aside, of the discrepancy in specificity described above, another natural quality that human languages have that affects machine learning vision-language systems regards their **subjectivity**. What Leo describes (or imagines) as a beautiful house by a lake, with all chance, will be drastically different from what Panos envisions in his mind with that description. The topic of how different agents use preexisting knowledge, or developed over the time of a discourse knowledge, *specific to their communication partner* (in context), to reduce their uncertainty and subjectivity, and make more effective and informed dialogues is broad [88, 99, 148, 133], fascinating but beyond the scope of this thesis. However, it is worth mentioning that **our recent work** at the intersection of emotions and linguistic explanations for visual art, ArtEmis [8], is the **first** visio-linguistic study that embraces the subjectivity of the meaning of words (and emotional reactions) when creating *neural-based speakers*, similar to those of Chapter 5.

1.1.5 Main research direction, contributions, and thesis outline

The fundamental research question behind this thesis concerns how to create semantically rich and robust representations of 3D objects that improve machine-based object-centric reasoning, with (or without) the inclusion of human referential language. Approximately half of the thesis is dedicated to building novel, deep-learning-based models for shape synthesis and generation that are unaware of any language and vary w.r.t. the richness of supervising signals they use. Ranging from self-supervised generative networks to networks that explicitly input semantic object parts or (implicitly) shape correspondences. The remaining half of the thesis puts referential language at the center of its study. The strong connection with the first half is that the explored referential language concerns (again) 3D objects and their properties, e.g., shape or spatial-location, and is discriminative. That is, the used language acts as an auxiliary signal that we can, and do use, to **discriminate** among **generative** object representations like those of the first half.

In summary:

1. Chapter 2 introduces AutoEncoders and (the first) Generative Adversarial Networks for 3D point clouds of common objects, and establishes valuable and widely adopted evaluation metrics for point cloud shape-synthesis at scale. These networks use **no** supervision and added a new way of creating latent object representations in the 3D deep learning literature.

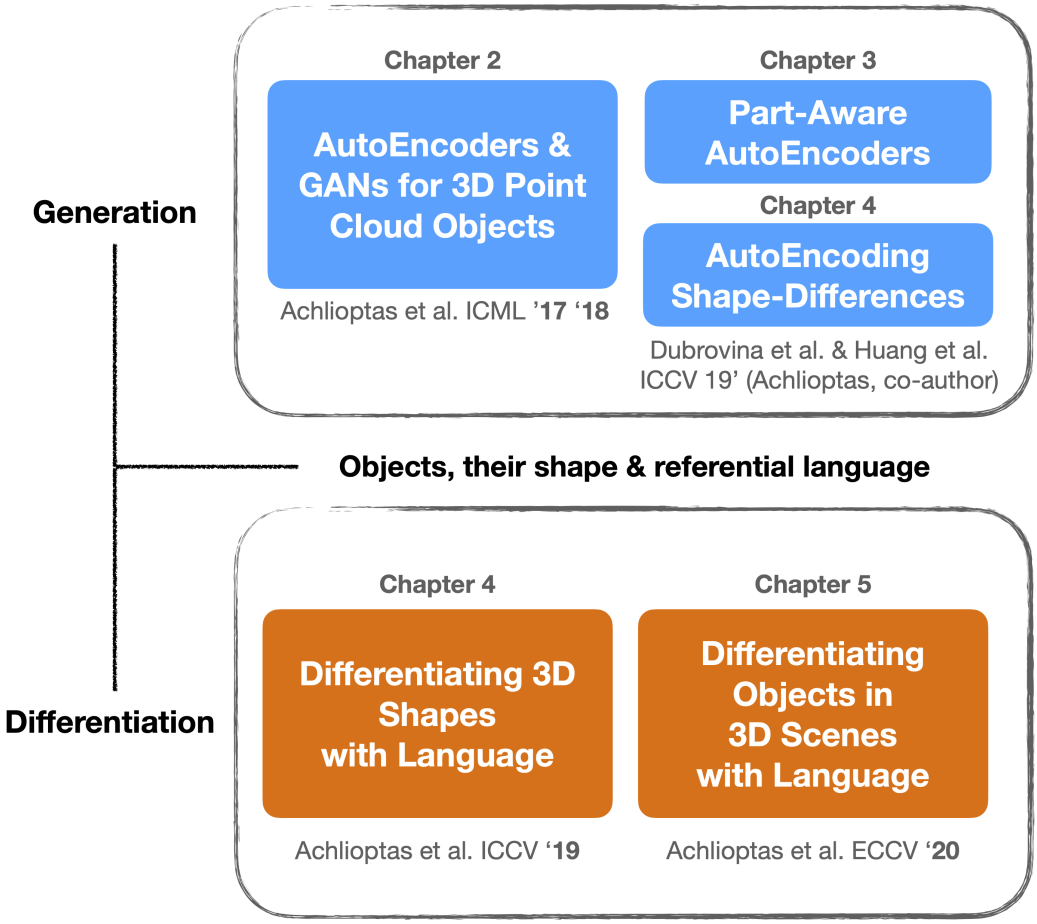


Figure 1.1: Main scientific contributions, and structure of the thesis. The first half is dedicated to **generative modeling of object shapes**, and the second half to **discriminative modeling of shapes or objects, with referential language.**

2. Chapter 3 introduces a generative network that explicitly takes into account the part-based composition of common 3D objects. By exploiting the supervision of semantic shape-parts, such as the legs or back of a chair, it creates a *factorized* latent space where novel applications for deep shape synthesis are possible. These include part-aware latent interpolations of objects, and latent mixing-&-matching of their parts to create novel objects.

3. Chapter 4 introduces our third generative network for shapes which unlike the previous two learns to reconstruct a shape based on how it is *shape-wise different* from a fixed "base" shape. To create such differentiating input signal our approach relies on having (typically) dense e.g., point-to-point correspondences among the underlying shapes. The careful incorporation of

this rich supervising signal improves the latent representations of Chapter 2 with regards to the quality of making latent-based interpolations and analogies for 3D shapes.

4. Chapter 5 includes the first published study concerning the design of neural-based systems: "listeners and speakers", that use language that refers, explicitly and **only**, the shape of common objects. Our resulting multi-modal networks exploit the shape-aware point cloud representations of Chapter 2 to improve their operational performance and among others to also make the **first** neural-speakers and listeners that can (optionally) work directly with 3D point cloud representations of the underlying objects. Interestingly, deep learning referential language for shapes gives naturally rise to part-aware latent representations (similar to those of Chapter 3) without however explicitly requiring extra labels for an object's semantic parts.

5. The final Chapter 6, extends Chapter 5 in a challenging real-world scenario. It builds neural-listeners that operate with 3D point clouds of objects, but when those latter are embedded **inside** a 3D scene of real-world room/apartment. This setup forces our agents to learn properties that go beyond an object's shape, and pay attention also to, say, its texture, or relative spatial location. As we show in this chapter, creating neural listeners that *jointly analyze* the objects of a scene/context (via graph-neural-nets) is pivotal for learning to identify the referred object.

It is worth noting that to establish the studies of Chapters 5 and 6 **we had to create two large scale datasets (ShapeGlot & ReferIt3d)** containing natural language made by thousands of humans who solved the underlying cognitive tasks that we deep learned with our neural agents. These datasets, along all the technical contributions presented in this thesis, and their corresponding implementations, are *publicly available, widely used, and have been the main subjects of peer-reviewed and published research*. For a pictorial overview of the thesis structure for the following chapters and our own scientific contributions, please see Figure 1.1.

1.2 Overview of Deep Learning 3D Shapes and Objects

This section provides a quick review of the rapidly growing area of learning how to synthesize and analyze collections of shapes of 3D objects with neural networks. It starts by highlighting some of the obstacles found specifically when doing **3D** deep-learning modeling, as opposed to applying 2D-based learning techniques; and continues by summarizing the main 3D representations and deep learning tools explored in *this* thesis.

1.2.1 Main bottlenecks behind deep shape synthesis & analysis

Deep 3D shape synthesis Encoding and reconstructing 3D shapes is a fundamental problem in computer graphics, computer vision and related fields. Unlike images, which enjoy a canonical representation, 3D shapes are encoded through a large variety of representations, such as point clouds, triangle meshes and volumetric data, to name a few. Perhaps even more importantly, 3D shapes may undergo a diverse set of transformations, ranging from rigid motions to complex non-rigid and articulated deformations, that impact these representations. The representation issues have become even more prominent in recent years with the advent of learning-based techniques, leading to a number of solutions for learning directly on geometric 3D data [41]. This is challenging as point clouds and meshes **lack the regular grid structure** exploited by convolutional architectures. In particular, the problem of representations that are well adapted for both shape analysis and especially shape synthesis remains difficult. For example, several methods for shape interpolation have been proposed by designing deep neural networks, including AutoEncoder architectures, to interpolate the latent vectors learned by such networks (e.g., [252] or those introduced by us in [5, 6]). Unfortunately, it is not clear to what extent the latent vectors lie in a linear vector space, and thus linear interpolation can sometimes lead to unrealistic intermediate shapes. Even more, typical approaches (e.g., [252, 5, 6]) apply linear interpolation and other semantic-manipulations on vectors that are derived from a *holistic/unstructured* latent space which does not offer strong guarantees on the disentanglement of the latents, or an easy way to manipulate a subset of them [163]. In fact, this last observation partially motivated our work on Chapter 3 which follows up our work on Chapter 2 (on unstructured latent spaces) and which builds an explicitly **factorized** latent space for shape-object parts.

Deep 3D shape analysis At the same time, several recent techniques aim at applying deep learning for shape analysis. One of the main challenges here is defining a meaningful notion on convolution, while **ensuring invariance** to basic transformations, such as rotations and translations. Several techniques have been proposed based on e.g., geometry images [253], volumetric [181, 282], point-based [219] and multi-view approaches [220], as well as, more recently intrinsic techniques that adapt convolution to curved surfaces [180, 36] (see also [41] for an overview), and even via toric covers [179], among many others. Despite this tremendous progress in the last few years, defining a shape representation that is compact, lends itself naturally to learning, while being invariant to the desired class of transformations (e.g., rigid motions) and not limited to a particular topology, remains a challenge.

As we will also show with our work [111] in Chapter 4 one can deep learn *matrices* that encode how 3D shapes relate to each other in the context of shape collection based on a priori known pairwise **correspondences** e.g., dense-pointwise maps. Such a matrix representation naturally leads to *multiplicative* algebra for shape interpolation with oftentimes improved results compared to the classic linear-latent approach (see Figure 4.6). More importantly, it enables *convolutional* deep learning due to its canonical structure. It is worth noticing however that this is not a free lunch: acquiring correspondences among shapes of a collection can be particularly hard, especially for non-isometric shapes such as those found in commonly used repositories like ShapeNet [46].

Deep analysis of objects inside 3D scenes The above paragraphs focus primarily on obstacles one faces when trying to apply deep learning to 3D objects that are treated as *independent entities*, e.g., deep learning a collection of 3D models in a manner similar to how most deep learning has been previously applied on collections of images ([139], [68]). However, unlike typical imagery, one can arrange 3D models inside an ambient space to emulate layouts of rooms, landscapes and generic scenes [96, 297]. Even more, one can acquire rich 3D signals portraying scenes that contain objects from the real-world [20, 63, 45]; and as of recently to attempt hard and novel deep learning tasks with them [295, 63, 4, 54]. Deep learning such object-rich, compositional scenes, adds an extra level of difficulty compared to analyzing objects independently and brings unique challenges. To narrow the scope we will describe here the relevant problems that arose in consideration to our work [4] which is fully explored in Chapter 6. Note that these problems are broad and applicable to a wide range of tasks humans regularly solve when interacting with their 3D environment [297, 172].

Concretely, assume that one wants to describe properties of a 3D object in a scene to enable its identification from a machine (the main task we teach machines in Chapter 6). The first sub-task that the machine would naturally have to be able to do to achieve this goal would be to **segment** the scene in its constituent objects [63, 218]. Such a task calls for **joint analysis** of the underlying objects of the scene. Unfortunately, despite many excellent recent efforts ([222, 218, 217]) our community does not yet have robust tools for solving this problem, and despite that graph-neural-networks [132, 285] offer an elegant solution that in theory can learn joint compute over 3D objects (represented as nodes of a graph); their operational performance is typically far from perfect [4, 54, 78, 209, 100]. Next, even if one assumes that the object segmentation of a scene is given, the perceptual task of classifying the **fine-grained** properties of the objects of a scene, is not solved [271, 306, 231, 316]. Treating the objects as being independent i.e., without any joint analysis seems wasteful since there are strong spatial-based correlations among the objects and their classes/properties. More importantly, one

can embed 3D objects in a scene in a myriad ways and without using any canonical orientation, or scale – exaggerating thus the transformation-related problems mentioned before (e.g., building rotation-invariant representations). To make things worse the available 3D-scene-related data are scarce, making data-hungry deep learning approaches, less successful and "plug-&-play". Finally, for problems involving an agent moving/navigating inside a scene while interacting with it, e.g., in their attempt to make a referential description for a given object; the **relative view** between the agent and the underlying object can have a strong effect to the output result (see Figure 6.5 for how it affects descriptions of objects). Making deep-learning-based representations for sets of objects that can reflect the (arbitrary) relative angle an external observer who is interacting with them might have, is an open problem.

1.2.2 3D representations and neural blocks used in this thesis

Before I present the upcoming technical chapters, I wish to give a brief summary on the *main* 3D representations and deep-learning tools we used in the research conducted **in this thesis**. This subsection will summarize such technical elements, without the ideas, execution-pipelines or motivation behind their use, and its intent is to provide only an overview of the *most basic* prerequisites one has to have to fully appreciate the upcoming chapters.

This thesis explores a *wide range* of 3D object representations In Chapter 2 we present our research on 3D representation learning concering 3D **point clouds** of common man-made objects (e.g., Figure 2.1), or to a lesser extent shapes of human bodies (e.g., Figures 2.7 and 2.6). Despite having a low-memory footprint and being widely used, point clouds lack a canonical grid structure and in the typical resolutions used in deep learning e.g., 2K-20K points per object, many thin and delicate object-structures e.g., decorative elements of chairs, are hardly captured by them. These facts make learning of fine-grained object properties, or high-frequency surface details, relatively hard with this modality.

Next, in Chapter 3 we turn our attention to **voxel-based** encodings of common objects. Voxelized representations come equipped with a regular 3D grid structure which enables us to use well-studied convolutional operators for encoding and decoding the underlying objects. It is worth noting however, that voxels have a high-memory footprint (cubic in cost) and in order to capture thin and delicate object-structures, one has to operate in high resolutions (e.g., 128^3 voxels) restricting our method's scalability compared to, say, a point-cloud-based alternative.

In Chapter 4 we use a different 3D object represenation to conduct our study. That of (triangular)

meshes. Meshes have high fidelity and can 'perfectly' capture the high-frequency details of their underlying shape. Meshes are also widely popular as they are regularly used by human designers, resulting in an abundance of mesh-encoded CAD models. Nevertheless, similar to point clouds, meshes also lack a canonical grid structure making convolutional deep learning with them less straightforward.

Finally, in Chapter 5, we explore a **combination** of two object representations. That is we design neural networks that operate with both 3D point clouds, and **2D renderings** of their underlying objects (see Figure 5.3). Using 2D image-renderings to portray 3D objects is not a new idea – in fact, the first deep classification network for 3D shapes operated on multi-view 2D renderings of CAD models ([293], circa 2015). Rendered images can portray high frequency content (limited though in a single view-point), and can also be directly used by established convolutional networks [251, 101] including networks pre-trained on large-scale data like ImageNet [68]. As we show with our study in Chapter 5 Table 5.2, there are significant performance benefits in building neural-listeners that have access to both 2D and 3D objects representations.

1.3 Overview of Tools Used for Deep Learning Language

In this section we give a short overview of some of the main deep-learning language oriented tools that we use in the works of Chapters 5 and 6. Such neural modules (e.g., an "LSTM") are fundamental and have nowadays become mainstream. Since, in the coming chapters some of their details are being opaque, and are treated as prior knowledge, for reasons of completeness we expose their basic details and properties here. We also include some relevant but not used by us techniques e.g., using feed-forward nets for language crunching to enrich the educational character of this section. We note that parts of this section are based on the excellent review: "Deep Learning Based Text Classification: A Comprehensive Review" [187] which provides a much broader exposition.

1.3.1 Feed-forward, recurrence and attention

Feed-forward neural networks for language processing Feed-forward networks are perhaps the simplest deep-learning models for processing linguistic data. The crucial simplifying assumption that enables feed-forward networks to operate on sequential linguistic data, is to altogether ignore the order between words and treat the input text as a **bag of words**. Typically, these models embed each word into a vector using a pre-trained word-embedding model like the Word2Vec [185] or Glove [207]. The resulting word-representations are typically being further fine-tuned under a

task-specific loss function, and are being aggregated with symmetric functions (symmetric in the order of their inputs) such as the sum or average pooling. The output result of applying such an aggregating function is finally passed through one or more feed-forward layers i.e., Multi-Layer Perceptrons (MLPs); of which the output constitutes the text's final representation. Surprisingly, such low-complexity models have achieved high accuracy on many text-classification benchmarks e.g., [114], and are considered a good baseline to access the extent to which word-order matters for a given problem, or to be used in cases where larger language models are not practically feasible. An intuitive extension of this syntax-unaware approach is to use a bag of *n-grams* as additional features to capture *local* word order information [117]. This turns out to be very efficient in practice, achieving comparable results to many methods that explicitly model the global word order [283].

Remark It is worth noting that the above neural-architecture 'recipe' was separately (re)-introduced in the point cloud processing community around 2017 with works like PointNet [219] (see also Chapter 2) and Deep Sets [317]. The equivalence between the two practices becomes apparent if one treats a single 3D point as a single word which can be embedded in a high-dimensional feature space (like the one provided by Word2Vec), and a point cloud as an order-less sentence, i.e., a set, for which a feature can be extracted by applying, like above, a symmetric function to its constituent members.

Modeling syntax recurrently Recurrent Neural Networks (RNNs) treat text more naturally than feed-forward networks, viewing it as a sequence of words, and are intended to capture long-range word dependencies and text structures. RNNs belong to a broad category of deep neural networks, known as recursive neural networks. A recursive neural network applies the *same weights recursively* over a structured input to produce a structured prediction or a vector representation over a *variable-size* input. In the context of processing linguistic data the most common text structure assumed is that of a linear chain, though in many applications more complex hierarchical structures, such as parse trees of natural language sentences are used [255]. Unfortunately, vanilla RNN models tend to not perform well, especially for longer sequences, and often underperform feed-forward neural networks. A primary reason behind this is that their optimization becomes unstable as the sequence length increases, suffering from vanishing or exploding gradients [206] [3]. Among the many variants of RNNs, the Long Short-Term Memory (LSTM) is perhaps the most widely used architecture – also used by us in our works of Chapters 5 and 6 to model referential sentences. LSTMs appear to attain strong empirical performance in a variety of text-related tasks [187] and effectively mitigate the

gradient problems suffered by vanilla RNNs. By design they incorporate an additional (unique to LSTMs) memory cell that can learn to remember *selectively* values over arbitrary length intervals, and three gates (input gate, output gate, forget gate) that overall regulate the flow of information in and out of the LSTM. Their capacity to forget irrelevant parts of the input effectively reduces the length of the input which in turn reduces the severity/instability of the underlying optimization. This ability of a neural-network to pay more or less attention to specific parts of its inputs relates to our next topic of "attention" which has drastically affected modern neural network designs [275].

Neural attention In the context of neural networks, neural attention is a technique that mimics the cognitive attention of humans and its main purpose is to concentrate the learning process and resources around the important parts of the input data while ignoring the rest. Neural attention has become an increasingly popular concept and a useful tool in developing deep learning models for a variety of language processing tasks [25, 171], visual tasks [190, 318], or both, in studies that involve learning simultaneously across the two modalities [8, 302, 13]. In a nutshell, attention can be interpreted as a vector of *importance weights*. These importance weights provide a ranking across a set of features, which can be the input words, pixels, etc. or higher-level abstractions computed by a deeper layer of a neural net. What is more important here, are the exact mechanics used to compute the weights and the specific structural bias these mechanics embed in the underlying architecture. The two most common basic techniques for creating attention values are the "dot-product" attention, which uses the dot product between vectors to determine their importance weights, and "multi-head" attention, which combines several attention mechanisms to direct the overall attention of a network or sub-network. Given a set of vectors $\mathcal{X} = \{x_1, x_2, \dots x_N\}$ corresponding to N input representations e.g., N words of a sentence, and a vector y that we assume encodes some critical aspect of the underlying learning task, one can use the dot-product attention to extract the importance-weight of each of the N inputs w.r.t. y, by applying $a_i \triangleq x_i^T \times y$, and as typically done normalize the corresponding values to a probability distribution by soft-maxing them: $\hat{a}_i = \frac{\exp(a_i)}{\sum_j^N \exp(a_j)}$. To see an application of this in the context of learning and attending to the important words of a referential sentence when solving **our shape identification task**, please read Section 5.5 and see output examples of such a process in Figure 5.4. Last, it is worth noting here that in many applications [275, 318] the above vector y is in fact the original set $\mathcal{X}$. The main idea behind such a case of "self-attention", is to discover the associations and correlations among the N inputs, which in turn can reveal the input values the network should pay attention to when processing its k-th input.

Chapter 2

AutoEncoders and GANs for 3D Point Cloud Objects

2.1 Overview

As we alluded to with our discussion in Section 1.1.1, three-dimensional geometric data portraying common objects offer an excellent domain for studying representation learning and generative modeling. In this chapter, we look at geometric data represented as point clouds. We introduce a deep AutoEncoder (AE) network with state-of-the-art reconstruction quality and generalization ability. The learned representations outperform existing methods on 3D recognition tasks and enable shape editing via simple algebraic manipulations, such as semantic part editing, shape analogies and shape interpolation, as well as shape completion. We perform a thorough study of different generative models including GANs operating on the raw point clouds, significantly improved GANs trained in the fixed latent space of our AEs, and Gaussian Mixture Models (GMMs). To quantitatively evaluate generative models we introduce measures of sample fidelity and diversity based on matching between sets of point clouds. Interestingly, our evaluation of generalization, fidelity and diversity reveals that GMMs trained in the latent space of our AEs yield the best results overall.

2.2 Introduction

Three-dimensional (3D) representations of real-life objects are a core tool for vision, robotics, medicine, augmented reality and virtual reality applications. Recent attempts to encode 3D geometry for use in deep learning include view-based projections, volumetric grids and graphs. In this chapter,

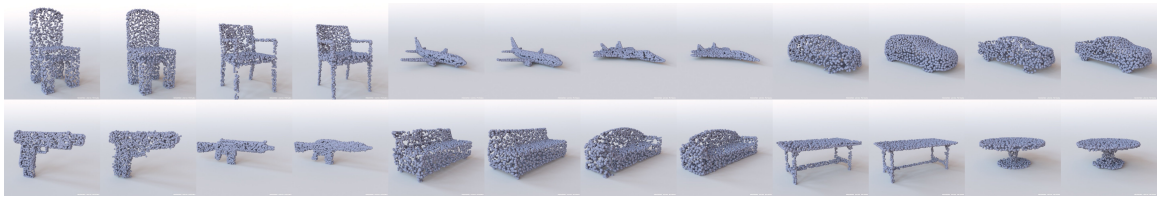


Figure 2.1: Reconstructions of unseen shapes from the test split of the input data. The leftmost image of each pair shows the ground truth shape, the rightmost the shape produced after encoding and decoding using a *class-specific* AE-EMD.

we focus on the representation of 3D point clouds. Point clouds are becoming increasingly popular as a homogeneous, expressive and compact representation of surface-based geometry, with the ability to represent geometric details while taking up little space. Point clouds are particularly amenable to simple geometric operations and are a standard 3D acquisition format used by range-scanning devices like LiDARs, the Kinect or iPhone's face ID feature.

All the aforementioned encodings, while effective in their target tasks (e.g., rendering or acquisition), are hard to manipulate directly in their raw form. For example, naïvely interpolating between two cars in any of those representations does not yield a representation of an "intermediate" car. Furthermore, these representations are not well suited for the design of generative models via classical statistical methods. Using them to edit and design new objects involves the construction and manipulation of custom, object-specific parametric models, that link the semantics to the representation. This process requires significant expertise and effort.

Deep learning brings the promise of a *data-driven approach*. In domains where data is plentiful, deep learning tools have eliminated the need for hand-crafting features and models. Architectures like AutoEncoders (AEs) [238, 131] and Generative Adversarial Networks (GANs) [87, 225, 48] are successful at learning data representations and generating realistic samples from complex underlying distributions. However, an issue with GAN-based generative pipelines is that training them is notoriously hard and unstable [240]. In addition, and perhaps more importantly, *there is no universally accepted method for the evaluation of generative models*.

In this chapter, we explore the use of deep architectures for *learning representations* and introduce the first deep *generative models* for *point clouds*. At the time this work appeared, only a handful of deep architectures tailored to 3D point clouds existed in the literature, and their focus was elsewhere: they either aimed at classification and segmentation [219, 222], or used point clouds *only* as an intermediate or output representation [118, 76]. Our specific contributions in this domain are:

- A new AE architecture for point clouds—inspired by recent architectures used for classification

[219]—that can learn compact representations with (i) good reconstruction quality on unseen samples; (ii) good classification quality via simple methods (SVM), outperforming the state of the art [290]; (iii) the capacity for meaningful semantic operations, interpolations and shape-completion.

- The first set of deep generative models for point clouds, able to synthesize point clouds with (i) measurably high fidelity to, and (ii) good coverage of both the training and the held-out data. One workflow that we propose is to first train an AE to learn a latent representation and then train a generative model in that fixed latent space. The GANs trained in the latent space, dubbed here *l-GANs*, are easier to train than raw GANs and achieve superior reconstruction and better coverage of the data distribution. Multi-class GANs perform almost on par with class-specific GANs when trained in the latent space.

- A study of various old and new point cloud metrics, in terms of their applicability (i) as reconstruction objectives for learning good representations; (ii) for the evaluation of generated samples. We find that a commonly used metric, Chamfer distance, fails to identify certain pathological cases.

- Fidelity and coverage metrics for generative models, based on an optimal matching between two different collections of point clouds. Our coverage metric can identify parts of the data distribution that are completely missed by the generative model, something that diversity metrics based on cardinality might fail to capture [21].

The rest of this chapter is organized as follows: Section 2.3 provides a quick overview of related work. Section 2.4 outlines some background for the basic building blocks of our work. Section 2.5 introduces our metrics for the evaluation of generative point cloud pipelines. Section 2.6 discusses our architectures for latent representation learning and generation. In Section 2.7, we perform comprehensive experiments evaluating all of our models both quantitatively and qualitatively. Further results can be found in the Appendix A. Last, the code for all our models is publicly available*.

2.3 Related Work

At the beginning of 3D deep learning At the time this work appeared (circa 2017), deep learning architectures operating on 3D objects were significantly less than nowadays where thousands of

*http://github.com/optas/latent_3d_points

papers exist on this topic. The main approaches involved deep learning architectures for view-based projections [258, 287, 118], volumetric grids [221, 293, 102], and graphs [42, 103, 66, 308]. A few works ([290], [286], [83], [39], [175], [325]) had explored generative and discriminative representations for geometry. Moreover, those operated on different modalities, typically voxel grids or view-based image projections. To the best of our knowledge, our work was the first to study generative representations for 3D point clouds.

Training generative networks The work of *Salimans et al.* [240] gives some useful techniques that help training a classic monolithic GAN. They propose feature matching as a method of regularizing the training process and ameliorating mode collapse issues. Variational autoencoders (VAEs) [131] learn a representation compatible with a given prior distribution. Their training is more stable than GANs, though the reported results seem to be of lower quality (i.e. slightly blurry images [86]). VAEGANs [146] improve upon VAEs by training a VAE jointly with a GAN. DCGANs and LAPGANs [225, 69] involve training of models of increasing level of detail, making training a bit smoother. MDGANs [48] introduce regularizers in the GAN objective meant to prevent mode collapse and improve the stability of training. WGANs [19] are a very elegant and intuitive solution to the problem of vanishing gradients and mode collapse. They optimize an objective based on the Wasserstein distance between the data and generator distributions, which is known to be more stable and yield good results.

All of the GAN variants above improve the vanilla GAN is some way, but all are nonetheless monolithic. Perhaps closer to our training workflow approach are the following results:

- The "plug and play" generative architecture [198] essentially uses some pretrained components, which allows for substituting a core "condition" network that guides a pretrained generator. This approach had achieved excellent results for image generation, though it involved a number of complex steps (e.g. Langevin sampling). Our proposed AE+GAN workflow is much simpler and seems to be a great fit for the point cloud modality.

- Adversarial autoencoders [176] use a GAN in the latent space of an AE. However they are still trained *jointly*.

- GMMNs [158] are an alternative to GANs. They only train a generator and use a moment-matching objective to bring the generator distribution close to the data distribution. One of the proposed variants involves training in the latent space of an already training AE, similarly to our approach, which significantly improves performance. They have enjoyed limited adoption,

likely due to the moment matching objective's computational complexity - quadratic on the number of samples. This limits their applicability in high dimensions.

- Older work on sampling from autoencoders [230, 31, 30].

It is also worth noting that training Gaussian mixture models (GMM) in the latent space of an autoencoder is closely related to VAEs. One documented issue with VAEs is over-regularization: the regularization term associated with the prior, is often so strong that reconstruction quality suffers [37, 256, 130, 73]. The literature contains methods that start only with a reconstruction penalty and slowly increase the weight of the regularizer.

2.4 Technical Background

In this section we give the necessary background on point clouds, their metrics and the fundamental building blocks that we will use in the rest of the chapter.

2.4.1 Point clouds

Definition A point cloud represents a geometric shape—typically its surface—as a set of 3D locations in a Euclidean coordinate frame. In 3D, these locations are defined by their x, y, z coordinates. Thus, the point cloud representation of an object or scene is a $N \times 3$ matrix, where N is the number of points, referred to as the point cloud resolution.

Point clouds as an input modality present a unique set of challenges when building a network architecture. As an example, the convolution operator—now ubiquitous in image-processing pipelines—requires the input signal to be defined on top of an underlying grid-like structure. Such a structure is not available in raw point clouds, which renders them significantly more difficult to encode than images or voxel grids. Recent classification work on point clouds (PointNet [219]) bypasses this issue by avoiding convolutions involving groups of points. Another related issue with point clouds as a representation is that they are permutation invariant: any reordering of the rows of the point cloud matrix yields a point cloud that represents the same shape. This property complicates comparisons between two point sets which is needed to define a reconstruction loss. It also creates the need for making the encoded feature permutation invariant.

Metrics Two permutation-invariant metrics for comparing unordered point sets have been proposed in the literature [76]. On the one hand, the *Earth Mover's* distance (EMD) [237] is the solution of

a transportation problem which attempts to transform one set to the other. For two equally sized subsets $S_1 \subseteq R^3, S_2 \subseteq R^3$, their EMD is defined by

$$d_{EMD}(S_1, S_2) = \min_{\phi: S_1 \to S_2} \sum_{x \in S_1} \|x - \phi(x)\|_2$$

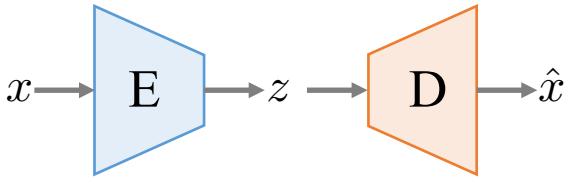
where ϕ is a bijection. As a loss, EMD is differentiable almost everywhere. On the other hand, the *Chamfer* (pseudo)-distance (CD) measures the squared distance between each point in one set to its nearest neighbor in the other set:

$$d_{CH}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2.$$

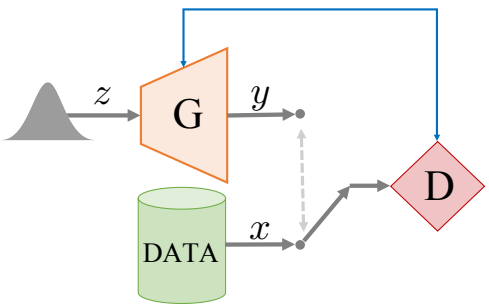
CD is differentiable and compared to EMD more efficient to compute.

2.4.2 Fundamental building blocks

AutoEncoders One of the main deep-learning components we use in this chapter is the *AutoEncoder* (AE, inset), which is an architecture that learns to reproduce its input. AEs can be especially useful, when they contain a narrow *bottleneck layer* between input and output. Upon successful training, this layer provides a low-dimensional representation, or *code*, for each data point. The Encoder (E) learns to compress a data point $\boldsymbol{x}$ into its latent representation, $\boldsymbol{z}$. The Decoder (D) can then produce a reconstruction $\hat{\boldsymbol{x}}$, of $\boldsymbol{x}$, from its encoded version $\boldsymbol{z}$.



Generative Adversarial Networks In this chapter we also work with Generative Adversarial Networks (GANs), which are state-of-the-art generative models. The basic architecture (inset) is based on a adversarial game between a *generator* (G) and a *discriminator* (D). The generator aims to synthesize samples that look indistinguishable from real data (drawn from $\boldsymbol{x} \sim p_{\text{data}}$) by passing a randomly drawn sample from a simple distribution $\boldsymbol{z} \sim p_z$ through the generator function. The discriminator is tasked



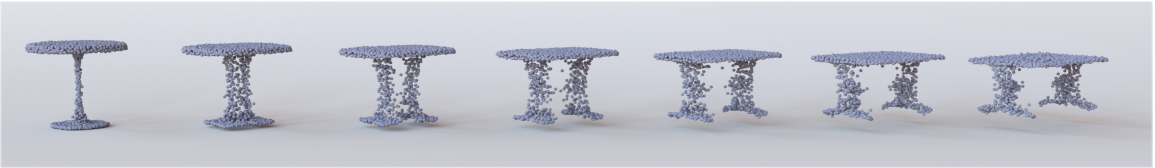


Figure 2.2: Interpolating between different point clouds, using our latent space representation. More examples for furniture and *human-form* objects [33] are demonstrated in Figures 2.4 and 2.6, respectively.

with distinguishing synthesized from real samples.

The most commonly losses used for the discriminator and generator networks are:

$$J^{(D)}(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}) = -\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log D(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{z}\sim p_z} \log \Big(1 - D\left(G(\boldsymbol{z})\right)\Big) , \tag{2.1}$$

$$J^{(G)}(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}) = -\mathbb{E}_{\boldsymbol{z}\sim p_z} \log D(G(\boldsymbol{z})) , \tag{2.2}$$

where $\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}$ are the parameters for the discriminator and the generator network respectively. In addition to the classical GAN formulation, we also use the improved Wasserstein GAN [94], which has shown improved stability during training.

Gaussian Mixture Model A GMM is a probabilistic model for representing a population whose distribution is assumed to be multimodal Gaussian, i.e. comprising of multiple subpopulations, where each subpopulation follows a Gaussian distribution. Assuming the number of subpopulations is known, the GMM parameters (means and variances of the Gaussians) can be learned from random samples, using the Expectation-Maximization (EM) algorithm [67]. Once fitted, the GMM can be used to sample novel synthetic samples.

2.5 Evaluation Metrics for 3D Generative Models

An important component of this work is the introduction of measures that enable comparisons between two sets of points clouds A and B. These metrics are useful for assessing the degree to which point clouds, synthesized or reconstructed, represent the same population as a held-out test set. Our three measures are described below.

JSD The Jensen-Shannon Divergence between marginal distributions defined in the Euclidean 3D space. Assuming point cloud data that are axis-aligned and a canonical voxel grid in the ambient space; one can measure the degree to which point clouds of A tend to occupy similar locations as those of B. To that end, we count the number of points lying within each voxel across *all* point clouds of A, and correspondingly for B and report the JSD between the obtained empirical distributions (P_A, P_B):

$$JSD(P_A \parallel P_B) = \frac{1}{2}D(P_A \parallel M) + \frac{1}{2}D(P_B \parallel M)$$

where $M = \frac{1}{2}(P_A + P_B)$ and $D(\cdot \parallel \cdot)$ the KL-divergence between the two distributions [143].

Coverage For each point cloud in A we first find its closest neighbor in B. Coverage is measured as the *fraction* of the point clouds in B that were matched to point clouds in A. Closeness can be computed using either the CD or EMD point-set distance of Section 2.4, thus yielding two different metrics, COV-CD and COV-EMD. A high coverage score indicates that most of B is roughly represented within A.

Minimum Matching Distance (MMD) Coverage does not indicate exactly *how well* the covered examples (point-clouds) are represented in set A; matched examples need not be close. We need a way to measure the *fidelity* of A with respect to B. To this end, we match every point cloud of B to the one in A with the minimum distance (MMD) and report the average of distances in the matching. Either point-set distance can be used, yielding MMD-CD and MMD-EMD. Since MMD relies directly on the distances of the matching, it correlates well with how faithful (with respect to B) elements of A are.

Discussion The complementary nature of MMD and Coverage directly follows from their definitions. The set of point clouds A captures all modes of B with good fidelity when MMD is small *and* Coverage is large. JSD is fundamentally different. First, it evaluates the similarity between A and B in coarser way, via marginal statistics. Second and contrary to the other two metrics, it requires pre-aligned data, but is also computationally friendlier. We have found and show experimentally that it correlates well with the MMD, which makes it an efficient alternative for e.g. model-selection, where one needs to perform multiple comparisons between sets of point clouds.

2.6 Models for Representation Learning and Generation of 3D Objects

In this section we describe the architectures of our neural networks starting from an autoencoder. Next, we introduce a GAN that works directly with 3D point cloud data, as well as a decoupled approach which first trains an AE and then trains a minimal GAN in the AE's latent space. We conclude with a similar but even simpler solution that relies on classical Gaussian mixtures models.

2.6.1 Learning representations of 3D point clouds

The input to our AE network is a point cloud with 2048 points (2048×3 matrix), representing a 3D shape. The encoder architecture follows the design principle of [219]: 1-D convolutional layers with kernel size 1 and an increasing number of features; this approach encodes every point *independently*. A "symmetric", permutation-invariant function (e.g., a max pool) is placed after the convolutions to produce a joint representation. In our implementation we use 5 1-D convolutional layers, each followed by a ReLU [195] and a batch-normalization layer [113]. The output of the last convolutional layer is passed to a feature-wise maximum to produce a k-dimensional vector which is the basis for our latent space. Our decoder transforms the latent vector using 3 fully connected layers, the first two having ReLUs, to produce a 2048×3 output. For a permutation invariant objective, we explore both the EMD approximation and the CD (Section 2.4) as our structural losses; this yields two distinct AE models, referred to as AE-EMD and AE-CD. To regularize the AEs we considered various bottleneck sizes, the use of drop-out and on-the-fly augmentations by randomly-rotating the point clouds. The effect of these choices is showcased in the Appendix (Section A.1) along with the detailed training/architecture parameters. In the remainder of the chapter, unless otherwise stated, we use an AE with a 128-dimensional bottleneck layer.

2.6.2 Generative models for point clouds

Raw point cloud GAN (r-GAN) Our first GAN operates on the raw 2048×3 point set input. The architecture of the discriminator is identical to the AE (modulo the filter-sizes and number of neurons), without any batch-norm and with leaky ReLUs [173] instead or ReLUs. The output of the last fully connected layer is fed into a sigmoid neuron. The generator takes as input a Gaussian noise vector and maps it to a 2048×3 output via 5 FC-ReLU layers.

Figure 2.3: Editing parts in point clouds using simple additive algebra on the AE latent space. Left to right: tuning the appearance of cars towards the shape of convertibles, adding armrests to chairs, removing handle from mug. Note that the height of chairs with armrests is on average 13% shorter than of chairs without one; which is reflected also in these results.

Latent-space GAN (l-GAN) For our l-GAN, instead of operating on the raw point cloud input, we pass the data through a pre-trained autoencoder, which is trained separately for each object class with the EMD (or CD) loss function. Both the generator and the discriminator of the l-GAN then operate on the bottleneck variables of the AE. Once the training of GAN is over, we convert a code learned by the generator into a point cloud by using the AE's decoder. Our chosen architecture for the l-GAN, which was used throughout our experiments, is *significantly* simpler than the one of the r-GAN. Specifically, an MLP generator of a single hidden layer coupled with an MLP discriminator of two hidden layers suffice to produce measurably good and realistic results.

Gaussian mixture model In addition to the l-GANs, we also fit a family of Gaussian Mixture Models (GMMs) on the latent spaces learned by our AEs. We experimented with various numbers of Gaussian components and diagonal or full covariance matrices. The GMMs can be turned into point cloud generators by first sampling the fitted distribution and then using the AE's decoder, similarly to the l-GANs.

2.7 Experimental Evaluation

In this section we experimentally establish the validity of our proposed evaluation metrics and highlight the merits of the AE-representation (Section 2.7.1) and the generative models (Section 2.7.4). In all experiments in the thesis, we use shapes from the ShapeNet repository [46], that are axis aligned and centered into the unit sphere. To convert these shapes (meshes) to point clouds we uniformly sample their faces in proportion to their area. Unless otherwise stated, we train models with point clouds from a single object class and work with train/validation/test sets of an 85%-5%-10% split. When reporting JSD measurements we use a 28^3 regular voxel grid to compute the statistics.

2.7.1 Representational power of the AE

We begin with demonstrating the merits of the proposed AE. First we report its generalization ability as measured using the MMD-CD and MMD-EMD metrics. Next, we utilize its latent codes to do semantically meaningful operations. Finally, we use the latent representation to train SVM classifiers and report the attained classification scores.

Generalization ability Our AEs are able to reconstruct unseen shapes with quality almost as good as that of the shapes that were used for training. In Fig. 2.1 we use our AEs to encode unseen samples from the *test* split (the left of each pair of images) and then decode them and compare them visually to the input (the right image). To support our visuals quantitatively, in Table 2.1 we report the MMD-CD and MMD-EMD between reconstructed point clouds and their corresponding ground-truth in the train and test datasets of the chair object class. The generalization gap under our metrics is small; to give a sense of scale for our reported numbers, note that the MMD is 0.0003 and 0.033 under the CD and EMD respectively between two versions of the test set that only differ by the randomness introduced in the point cloud sampling. Similar conclusions regarding the generalization ability of the AE can be made based on the reconstruction loss attained for each dataset (train or test) which is shown in Fig. A.1 of the Appendix A.

AE loss	MMD-CD		MMD-EMD	
	Train	Test	Train	Test
CD	**0.0004**	**0.0012**	0.068	0.075
EMD	0.0005	0.0013	**0.042**	**0.052**

Table 2.1: Generalization of AEs as captured by MMD. Measurements for reconstructions on the training and test splits for an AE trained with either the CD or EMD loss and data of the chair class; Note how the MMD favors the AE that was trained with the same loss as the one used by the MMD to make the matching.

Latent space and linearity Another argument against under/over-fitting can be made by showing that the learned representation is amenable to intuitive and semantically rich operations. As it is shown in several recent works, well trained neural-nets learn a latent representation where additive linear algebra works to that purpose [186, 265]. By linearly interpolating between the latent representations of two shapes and decoding the result we obtain intermediate variants between the two shapes. This produces a "morph-like" sequence with the two shapes at its end points as shown in Fig. 2.2 and Fig. 2.4). Our latent representation is powerful enough to support removing and merging shape parts,

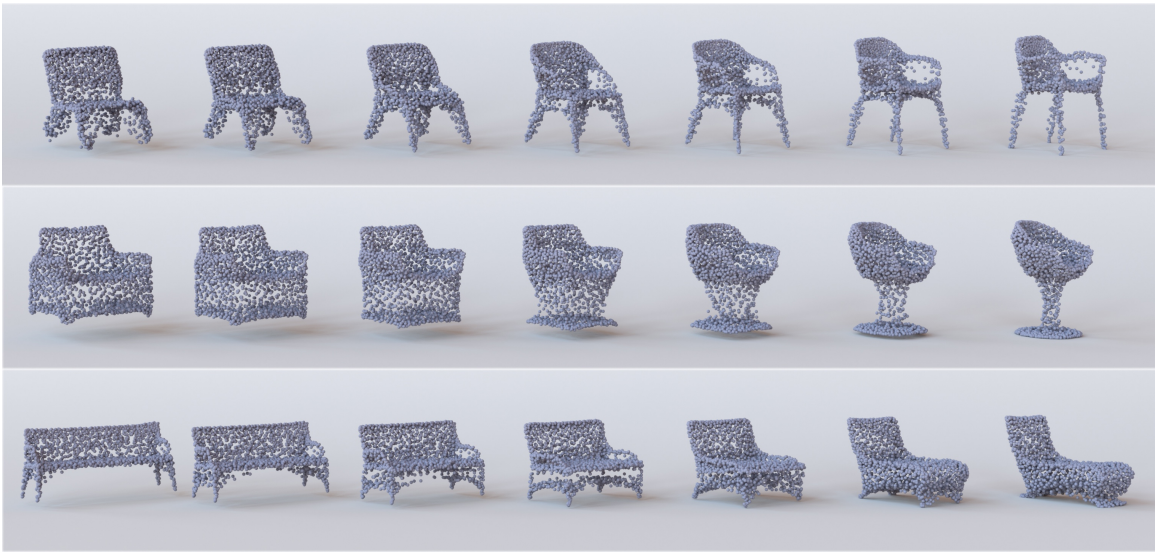


Figure 2.4: Interpolating between different point clouds (left and right-most of each row), using our latent space representation. Note the interpolation between structurally and topologically different shapes. **Note**: for all our illustrations that portray point clouds we use the Mitsuba renderer [116].

which enables morphing between shapes of significantly different appearance. Our cross-category latent representation enables morphing between shapes of different classes, cfg. the third row of Fig. 2.4 for an interpolation between a bench and a sofa.

Editing shape parts In Figure 2.3 we alter the input geometry (left) by adding, in latent space, the mean vector of geometries with a certain characteristic (e.g., convertible cars or cups without handles). For such a shape editing application, we use the embedding we learned with the AE-EMD trained *across all* 55 object classes, not separately per-category. This showcases its ability to encode features for different shapes, and enables interesting applications involving different kinds of shapes. Specifically, we use the shape annotations of Yi et al.[307] as guidance to modify shapes. As an example, assume that a given object category (e.g., chairs) can be further subdivided into two sub-categories $\mathcal{A}$ and $\mathcal{B}$: every object $A \in \mathcal{A}$ possesses a certain structural property (e.g., has armrests, is four-legged, etc.) and objects $B \in \mathcal{B}$ do not. Using our latent representation we can model this structural difference between the two sub-categories by the difference between their average latent representations $\mathbf{x}_{\mathcal{B}} - \mathbf{x}_{\mathcal{A}}$, where $\mathbf{x}_{\mathcal{A}} = \sum_{A \in \mathcal{A}} \mathbf{x}_A$, $\mathbf{x}_{\mathcal{B}} = \sum_{B \in \mathcal{B}} \mathbf{x}_B$. Then, given an object $A \in \mathcal{A}$, we can change its property by transforming its latent representation: $x_{A'} = x_A + \mathbf{x}_{\mathcal{B}} - \mathbf{x}_{\mathcal{A}}$, and decode $\mathbf{x}_{\mathcal{A}'}$ to obtain $A' \in \mathcal{B}$.

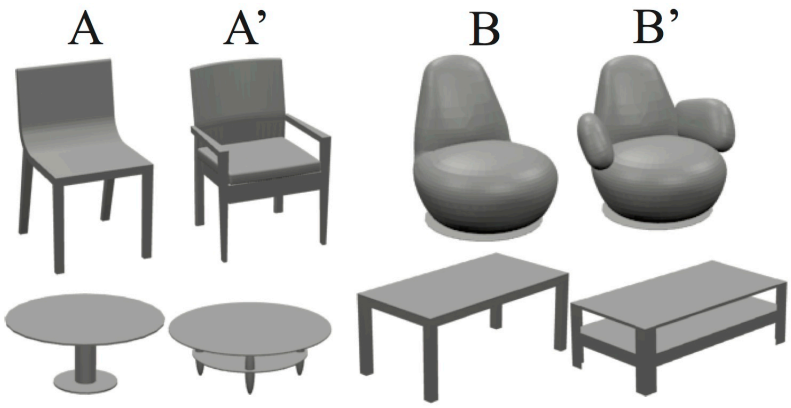


Figure 2.5: Shape analogies using our learned representation. Shape B' relates to B in the same way that shape A' relates to A.

Shape analogies Another demonstration of the Euclidean nature of the latent space is demonstrated by finding "analogous" shapes by a combination of linear manipulations and Euclidean nearest-neighbor searching. Concretely, we find the difference vector between A and A', we add it to shape B and search in the latent space for the nearest-neighbor of that result, which yields shape B'. We demonstrate the finding in Fig. 2.5 with images taken from the meshes used to derive the underlying point clouds to help the visualization. Finding shape analogies has been of interest recently in the geometry processing community [239, 109].

	A	B	C	D	E	ours EMD	ours CD
MN10	79.8	79.9	-	80.5	91.0	**95.4**	**95.4**
MN40	68.2	75.5	74.4	75.5	83.3	84.0	**84.5**

Table 2.2: Classification performance (in %) on ModelNet10/40. Comparing to A: SPH [127], B: LFD [49], C: T-L-Net [83], D: VConv-DAE [245], E: 3D-GAN [290].

Classification Our final evaluation for the AE's design and efficacy is done by using the learned latent codes as features for classification. For this experiment to be meaningful, we train an AE across all different shape categories: using 57,000 models from 55 categories of man-made objects. Exclusively for this experiment, we use a bottleneck of 512 dimensions and apply random rotations to the input point clouds along the gravity axis. To obtain features for an input 3D shape, we feed its point cloud into the AE and extract the bottleneck activation vector. This vector is then classified by a linear SVM trained on the de-facto 3D classification benchmark of ModelNet [293]. Table 2.2 shows comparative results. Remarkably, in the ModelNet10 dataset, which includes classes (chairs,

beds etc.) that are populous in ShapeNet, our simple AE significantly outperforms the state of the art [290] which instead uses several layers of a GAN to derive a 7168-long feature. In Fig. A.2 of the Appendix we include the confusion matrix of the classifier evaluated on our latent codes on ModelNet40 – the confusion happens between particularly similar geometries: a dresser vs. a nightstand or a flower-pot vs. a plant. The nuanced details that distinguish these objects may be hard to learn without stronger supervision.

Figure 2.6: Interpolating between different point clouds from the *test* split (left and right-most of each row) of the D-FAUST dataset of [33]. These linear interpolations have captured some of the dynamics of the corresponding motions: 'chicken-wings' (first row), 'shake shoulders' (second row) and 'jumping jacks' (third row). Compare to Fig. 2.7 that contains ground-truth point clouds in the same time interval.

2.7.2 Autoencoding human forms

In addition to ShapeNet core which contains man-made only objects, we have experimented with the D-FAUST dataset of [33] that contains meshes of human subjects. Specifically, D-FAUST contains 40K scanned meshes of 10 human subjects performing a variety of motions. Each human performs a set of (maximally) 14 motions, each captured by a temporal sequence of ~300 meshes. For our purposes, we use a random subset of 80 (out of the 300) meshes for each human/motion and extract from each mesh a point cloud with 4096 points. Our resulting dataset contains a total of 10240 point clouds and we use a train-test-val split of $[70\%, 20\%, 10\%]$ - while enforcing that every split contains *all* human/motion combinations. We use this data to train and evaluate an AE-EMD that is identical to the single-class AE presented in the thesis, with the only difference being the number of neurons

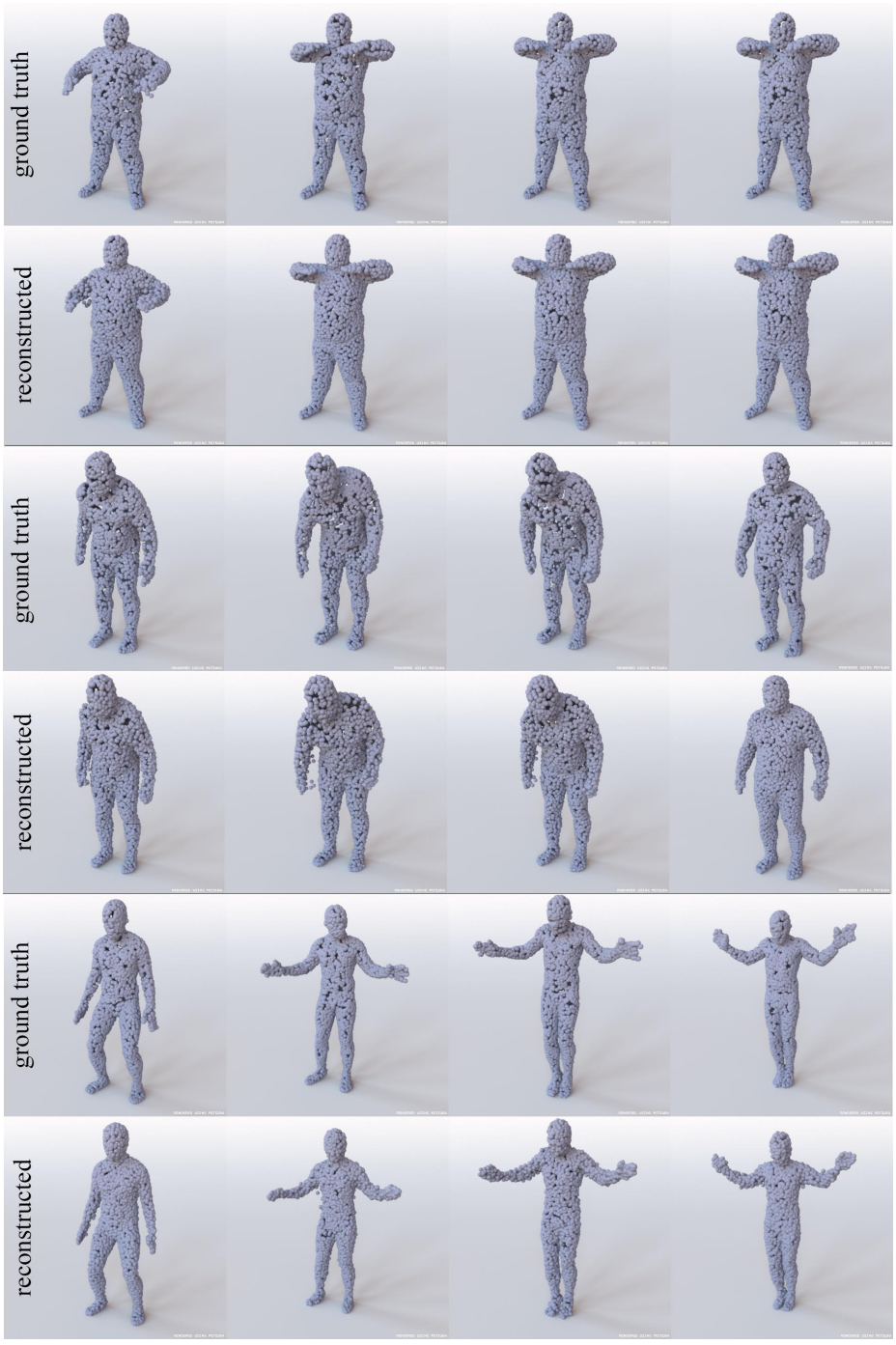


Figure 2.7: Reconstructions of unseen shapes from the *test* split extracted from the D-FAUST dataset of [33] with an AE-EMD decoding point clouds with 4096 points. In each row the poses depict a motion (left-to-right) as it progress in time.

of the last layer (4096×3 instead of 2048×3).

We demonstrate reconstruction and interpolation results in Figs. 2.7 and 2.6. For a given human subject and a specific motion we pick at random two meshes corresponding to time points t_0, t_1 (with $t_1 > t_0$) and show their reconstructions along with the ground truth in Fig. 2.7 (left-most and right-most of each row). In the same figure we also plot the reconstructions of two random meshes captured in (t_0, t_1) (middle-two of each row). In Fig. 2.6, instead of encoding/decoding the ground truth test data, we show decoded linear *interpolations* between the meshes of t_0, t_1.

Figure 2.8: Point cloud *completions* of a network trained with partial and complete (input/output) point clouds and the EMD loss. Each triplet shows the partial input from the test split (left-most), followed by the network's output (middle) and the complete ground-truth (right-most).

2.7.3 Shape completions

An important application that our AE architecture can be used for is that of completing point clouds that contain limited information of the underlying geometry. Typical range scans acquired for an object in real-time can often miss entire regions of the object due to the existence of self-occlusions and the lack of adequate (or "dense") view-point registrations. This fact makes any sensible solution to this problem of high practical importance. To address it here, we resort in a significantly different dataset than the ones used in the rest of this chapter. Namely, we utilize the dataset of [64] that contains pairs of complete (intact) 3D CAD models and *partial* versions of them. Specifically, for each object of ShapeNet (core) it contains six partial point clouds created by the aggregation of frames taken over a limited set of view-points in a virtual trajectory established around the object. Given this data, we first fix the dimensionality of the partial point clouds to be 2048 points for each one by randomly sub-sampling them. Second, we apply uniform-in-area sampling to each complete CAD model to extract from it 4096 points to represent a "complete" ground-truth datum. All the resulting point clouds are centered in the unit-sphere and (within a class) the partial and complete point clouds are co-aligned. Last, we train class-specific neural-nets with Chair, Table and Airplane data and a train/val/test split of [80%, 5%, 15%].

Architecture The high level design of the architecture we use for shape-completions is identical to the AE, i.e. independent-convolutions followed by FCs, trained under a structural loss (CD or EMD). However, essential parts of this network are different: depth, bottleneck size (controlling compression ratio) and the crucial differentiation between the input and the output data. Technically, the resulting architecture is an Abstractor-Predictor (AP) and is comprised by three layers of independent per-point convolutions, with filter sizes of $[64, 128, 1024]$, followed by a max-pool, which is followed by an FC-ReLU (1024 neurons) and a final FC layer (4096×3 neurons). We don't use batch-normalization between any layer and train each class-specific AP for a maximum of 100 epochs, with Adam, initial learning rate of 0.0005 and a batch size of 50. We use the minimal per the validation split model (epoch) for evaluating our models with the test data.

Evaluation We use the specialized point cloud completion metrics introduced in [262]. That is a) the *accuracy*: which is the fraction of the predicted points that are within a given radius (ρ) from any point in the ground truth point cloud and b) the *coverage*: which is the fraction of the ground-truth points that are within ρ from any predicted point. In Table 2.3 we report these metrics (with a $\rho = 0.02$ similarly to [262]) for class-specific networks that were trained with the EMD and CD losses respectively. We observe that the CD loss gives rise to more accurate but also less complete outputs, compared to the EMD. This highlights again the greedy nature of CD – since it does not take into account the matching between input/output, it can get generate completions that are more concentrated around the (incomplete) input point cloud. Figure A.6 shows the corresponding completions of those presented in Figure 2.8, but with a network trained under the CD loss.

Class	Airplane	Chair	Table
Test-size	4.5K	6K	6K
Acc-CD	**96.9**	**86.5**	**87.6**
Acc-EMD	94.7	77.1	78.4
Cov-CD	96.6	77.5	75.2
Cov-EMD	**96.8**	**82.6**	**83.0**

Table 2.3: Performance of point cloud *completions* on ShapeNet *test* data. Comparison between Abstractor-Predictors trained under the CD or EMD losses, on mean **Acc**uracy and **Cov**erage, across each class. The size of each test-split is depicted in the first row.

Figure 2.9: Synthetic point clouds generated by samples produced with l-GAN (top) and 32-component GMM (bottom), both trained on the latent space of an AE using the EMD loss.

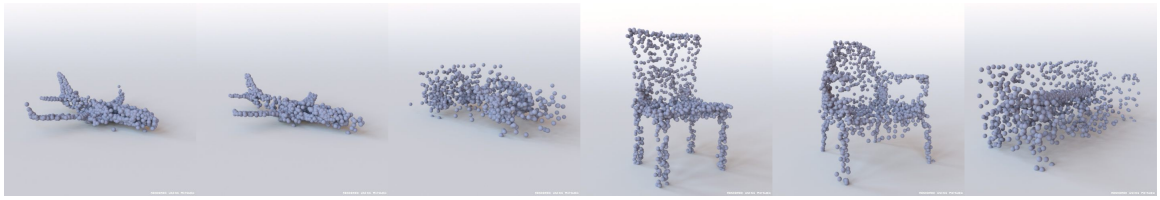


Figure 2.10: Synthetic results produced by the r-GAN. From left to right: airplanes, car, chairs, sofa.

2.7.4 Evaluating the generative models

Having established the quality of our AE, we now demonstrate the merits and shortcomings of our generative pipelines and establish one more successful application for the AE's learned representation. First, we conduct a comparison between our generative models followed by a comparison between our latent GMM generator and the state-of-the-art 3D voxel generator. Next, we describe how Chamfer distance can yield misleading results in certain pathological cases that r-GANs tends to produce. Finally, we show the benefit of working with a pre-trained latent representation in multi-class generators.

Comparison of our different generative models For this study, we train five generators with point clouds of the *chair* category. First, we establish two AEs trained with the CD or EMD loss respectively—referred to as AE-CD and AE-EMD and train an l-GAN in each latent space with the non-saturating loss introduced by *Goodfellow et al.* [87]. In the space learned by the AE-EMD we train two additional models: an identical (architecture-wise) l-GAN that utilizes the Wasserstein objective with gradient-penalty [94] and a family of GMMs with a different number of means and structures of covariances. We also train an r-GAN directly on the point cloud data.

Fig. 2.11 shows the JSD (left) and the MMD and Coverage (right) between the produced synthetic datasets and the held-out *test* data for the GAN-based models, as training proceeds. Note that the r-GAN struggles to provide good coverage and good fidelity of the test set; which alludes to the well-established fact that end-to-end GANs are generally difficult to train. The l-GAN (AE-CD)

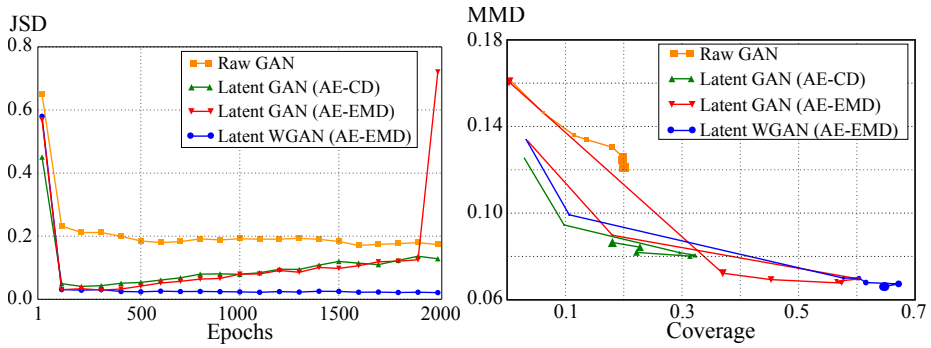


Figure 2.11: Learning behavior of the GANs, in terms of coverage / fidelity to the ground truth **test** dataset. Left – the JSD distance between the ground truth test set and synthetic datasets generated by the GANs at various epochs of training. Right – EMD based MMD/Coverage: curve markers indicate epochs 1, 10, 100, 200, 400, 1000, 1500, 2000, with *larger symbols denoting later epochs.*

performs better in terms of fidelity with much less training, but its coverage remains low. Switching to an EMD-based AE for the representation and otherwise using the same latent GAN architecture (l-GAN, AE-EMD), yields a dramatic improvement in coverage and fidelity. Both l-GANs though suffer from the known issue of mode collapse: half-way through training, first coverage starts dropping with fidelity still at good levels, which implies that they are overfitting a small subset of the data. Later on, this is followed by a more catastrophic collapse, with coverage dropping as low as 0.5%. Switching to a latent WGAN largely eliminates this collapse, as expected.

In Table 2.4, we report measurements for all generators based on the epoch (or underlying GMM parameters) that has minimal JSD between the generated samples and the validation set. To reduce the sampling bias of these measurements each generator produces a set of synthetic samples that is $3\times$ the population of the comparative set (test or validation) and repeat the process 3 times and report the averages. The GMM selected by this process has 32 Gaussians and a full covariance. As shown in Fig. A.4, GMMs with full covariances perform much better than those that have diagonal structure and $\sim$20 Gaussians suffice for good results. Last, the first row of Table 2.4 shows a baseline model that memorizes a random subset of the training data of the same size as the other generated sets.

Discussion. The results of Table 2.4 agree with the trends shown in Fig. 2.11 and further verify the superiority of the latent-based approaches and the relative gains of using an AE-EMD vs. an AE-CD. Moreover they demonstrate that a simple GMM can achieve results of comparable quality to a latent WGAN. Lastly, it is worth noting how the GMM has achieved similar fidelity as that of the perfect/memorized chairs and with almost as good coverage. Additional results in Appendix A show the same performance-based conclusions when our metrics are evaluated on the *train* split.

Model	Type	JSD	MMD-CD	MMD-EMD	COV-EMD	COV-CD
A	MEM	0.017	0.0018	0.063	78.6	79.4
B	RAW	0.176	0.0020	0.123	19.0	52.3
C	CD	0.048	0.0020	0.079	32.2	59.4
D	EMD	0.030	0.0023	0.069	57.1	59.3
E	EMD	0.022	0.0019	0.066	66.9	67.6
F	GMM	**0.020**	**0.0018**	**0.065**	**67.4**	**68.9**

Table 2.4: Evaluating 5 generators on the *test* split of the chair dataset on epochs/models selected via minimal JSD on the validation-split. We report: A: sampling-based memorization baseline, B: r-GAN, C: l-GAN (AE-CD), D: l-GAN (AE-EMD) , E: l-WGAN (AE-EMD), F: GMM (AE-EMD).

Chamfer's blindness, r-GAN's hedging An interesting observation regarding r-GAN can be made in Table 2.4. The JSD and the EMD based metrics strongly favor the latent-approaches, while the Chamfer-based ones are much less discriminative. To decipher this discrepancy we did an extensive qualitative inspection of the r-GAN samples and found many cases of point clouds that were over-populated in locations, that on average, *most* chairs have mass. This hedging of the r-GAN is particularly hard for Chamfer to penalize since one of its two summands can become significantly small and the other can be only moderately big by the presence of a few sparsely placed points in the non-populated locations. Figure 2.12 highlights this point. For a ground-truth point cloud we retrieve its nearest neighbor, under the CD, in synthetically generated sets produced by the r-GAN and the l-GAN and in-image numbers report their CD and EMD distances from it. Notice how the CD fails to distinguish the inferiority of the r-GAN samples while the EMD establishes it. This blindness of the CD metric to only partially good matches, has the additional side-effect that the CD-based coverage is consistently bigger than the EMD-based one.

Class	Fidelity		Coverage	
	A	Ours	A	Ours
car	0.059	**0.041**	28.6	**65.3**
rifle	0.051	**0.045**	69.0	**74.8**
sofa	0.077	**0.055**	52.5	**66.6**
table	0.103	**0.061**	18.3	**71.1**

Table 2.5: Fidelity (MMD-EMD) and coverage (COV-EMD) comparison between A: [290] and our GMM generative model on the *test* split of each class. Note that Wu et al. uses *all* models of each class for training contrary to our generators.

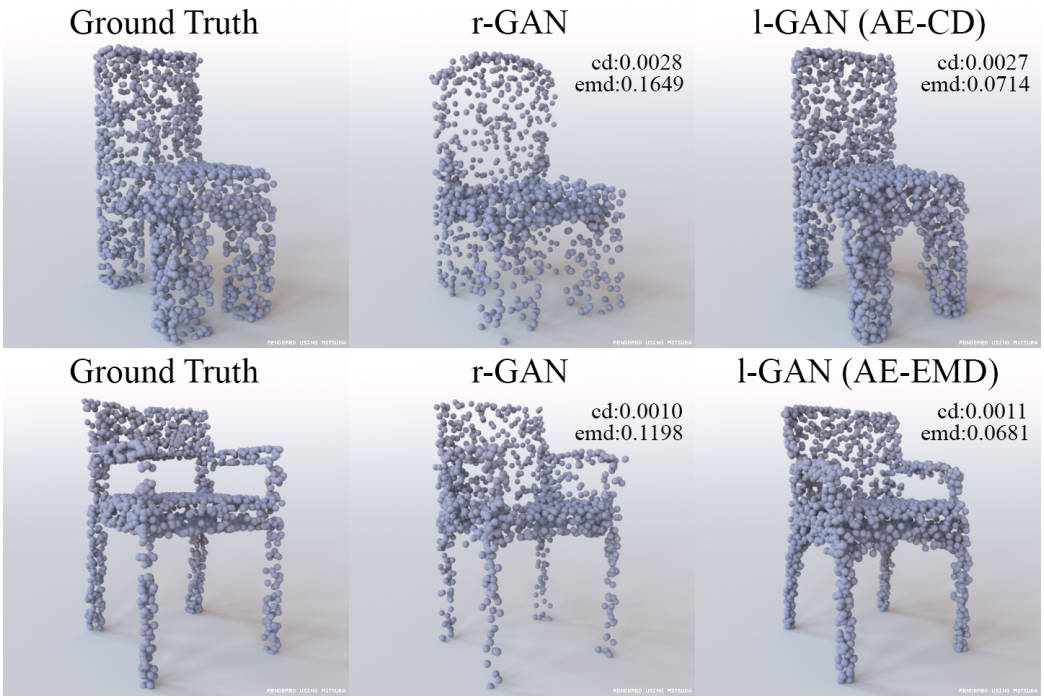


Figure 2.12: The CD distance is less faithful than EMD to visual quality of synthetic results; here, it favors r-GAN results, due to the overly high density of points in the seat part of the synthesized point sets.

Figure 2.13: Synthetic point clouds produced with l-WGANs trained in the latent space of an AE-EMD trained on a *multi-class* dataset.

Comparisons to voxel generators Generative models for other 3D modalities, like voxels, were also recently proposed [290]. One interesting question is: if point clouds are our target modality, does it make sense to use voxel generators and then convert to point clouds? This experiment answers this question in the negative. First, we make a comparison using a latent GMM which is trained in conjunction with an AE-EMD. Secondly, we build an AE which operates with *voxels* and fit a GMM in the corresponding latent space. In both cases, we use 32 Gaussians and a full covariance matrix for these GMMs. To use our point-based metrics, we convert the output of [290] and our voxel-based GMM into meshes which we sample to generate point clouds. To do this conversion we use the marching-cubes [152] algorithm with an isovalue of 0.1 for the former method (per authors' suggestions) and 0.5 for our voxel-AE. We also constrain each mesh to be a single connected

component as the vast majority of ground-truth data are.

Table 2.5 reveals how our point-based GMM trained with a class specific AE-EMD fares against [290] on four object classes for which the authors have made their (also class-specific) models publicly[†] available. Our approach is consistently better, with a coverage boost that can be as large as $4\times$ and an almost $2\times$ improved fidelity (case of table). This is despite the fact that [290] uses *a*ll models of each class for training, contrary to our generators that never had access to the underlying test split.

Table 2.6 reveals the performance achieved by pre-training a *voxel*-based AE for the chair class. Observe how by working with a voxel-based latent space, aside of making comparisons more direct to [290] (e.g., we both convert output voxels to meshes), we also establish significant gains in terms of coverage and fidelity.

	MMD-CD	MMD-EMD	COV-CD	COV-EMD
A	0.0046	0.091	19.6	22.4
Ours	**0.0025**	**0.072**	**60.3**	**64.8**

Table 2.6: MMD and Coverage metrics evaluated on the output of voxel-based methods at resolution 64^3, matched against the chair *test* set, using the same protocol as in Table2.4. Comparing: A: "raw" 64^3-voxel GAN [290] and a latent 64^3-voxel GMM.

Qualitative results In Fig. 2.9, we show some synthetic results produced by our l-GANs and the 32-component GMM. We notice high quality results from either model. The shapes corresponding to the 32 means of the Gaussian components can be found in the Fig. A.5, as well as results using the r-GAN (Fig.2.10).

	airplane	*car*	*chair*	*sofa*	*table*	average	multi-class
Tr	0.0004	0.0006	0.0015	0.0011	0.0013	**0.0010**	0.0011
Te	0.0006	0.0007	0.0019	0.0014	0.0017	**0.0013**	0.0014

Table 2.7: MMD-CD measurements for l-WGANs trained on the latent spaces of dedicated (left 5 columns) and multi-class EMD-AEs (right column). Also shown is the weighted average of the per-class values, using the number of train (Tr) resp. test (Te) examples of each class as weights. All l-WGANs use the model parameter resulted by 2000 epochs of training.

[†] http://github.com/zck119/3dgan-release

Multi-class generators Finally, we compare between class specific and class agnostic generators. In Table 2.7 we report the MMD-CD for l-WGANs trained in the space of either a dedicated (per-class) AE-EMD or with an AE-EMD trained with all listed object classes. It turns out that the l-WGANs produce perform similar results in either space. Qualitative comparison (Fig. 2.13) also reveals that by using a multi-class AE-EMD we do not sacrifice much in terms of visual quality compared to the dedicated AEs.

Figure 2.14: The AEs might fail to reconstruct uncommon geometries or might miss high-frequency details: first four images - left of each pair is the input, right the reconstruction. The r-GAN may synthesize noisy/unrealistic results, cf. a car (right most image).

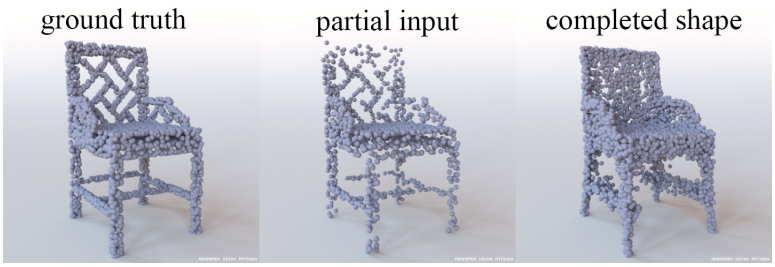


Figure 2.15: Our completion network might fail to preserve some of the style information in the partial point cloud, even though a reasonable shape is produced.

2.7.5 Limitations

Figure 2.14 shows some failure cases of our models. Chairs with rare geometries (left two images) are sometimes not faithfully decoded. Additionally, the AEs may miss high-frequency geometric details, e.g. a hole in the back of a chair (middle), thus altering the style of the input shape. Finally, the r-GAN often struggles to create realistic-looking shapes (right) – while the r-GAN chairs are easily visually recognizable, it has a harder time on cars. A limitation of our shape-completion pipeline regards the style of the partial shape, which might not be well preserved in the completed point cloud (see Fig. 2.15 for an example).

2.8 Conclusion

In this chapter we presented a novel set of architectures for 3D point cloud representation learning and generation. Our results show good generalization to unseen data and our representations encode meaningful semantics. In particular our generative models are able to produce faithful samples and cover most of the ground truth distribution. Interestingly, our extensive experiments show that the best generative model for point clouds is a GMM trained in the fixed latent space of an AE. While this might not be a universal result, it suggests that simple classic tools should not be dismissed. A thorough investigation on the conditions under which simple latent GMMs are as powerful as adversarially trained models would be of significant interest for future work. It is also worth pointing out that the materials of this chapter ([5, 6]) have been used and cited numerous times, including several works that attempt to improve 3D point cloud deep-learning based generative networs (e.g., [321, 157, 56, 51, 260, 281]). While most of these approaches improve certain aspects of this problem, overall creating realistically-looking point cloud models covering all variations existing in an underlying collection i.e., without suffering mode-collapse, has not been attained for objects of rich collections like ShapeNet. Thin structures, such as decorative elements, fine-grained surface details and overal rare shapes, are still hard to synthetically reproduce with high fidelity – keeping the 3D generation problem highly relevant and active.

Chapter 3

Building a Part-aware Latent Space for 3D Shapes

3.1 Overview

As discussed in Chapter 1 one of the main properties of objects is their part-based compositional structure. While the methods presented in Chapter 2 are very effective in creating meaningful latent object representations, they treat objects **holistically**, without any explicit modeling of their parts. Similarly, the results in Figure 2.3, show how one can use part-labels of shapes and simple linear algebra to edit a shape (e.g., add an armrest) by using a part-unaware latent space; and while they are promising, they are very from ideal. In this section, **we put the part-based structure of objects in the front-stage** of our research focus. Specifically, we develop a novel neural network architecture, termed *Decomposer-Composer*, that learns part-based, structure-aware shape modeling for objects represented as **3D voxel grids**. Our method utilizes an AutoEncoder-based pipeline, and produces a *factorized latent space*, where the semantic structure of the shape collection translates into a data-dependent sub-space factorization, and where shape composition and decomposition become simple linear operations on the embedding coordinates. We also propose to model shape assembly using an explicit learned part deformation module, which utilizes a 3D spatial transformer network to perform an in-network volumetric grid deformation, and which allows us to train the whole system end-to-end. As we show in Section 3.4, the resulting network allows us to perform part-level shape manipulation, unattainable by previously existing approaches, including those presented in Chapter 2. A high-level visual summary of the proposed key idea is shown in Figure 3.1. The materials of this section are based on our paper *Composite Shape Modeling via Latent Space Factorization* [75].

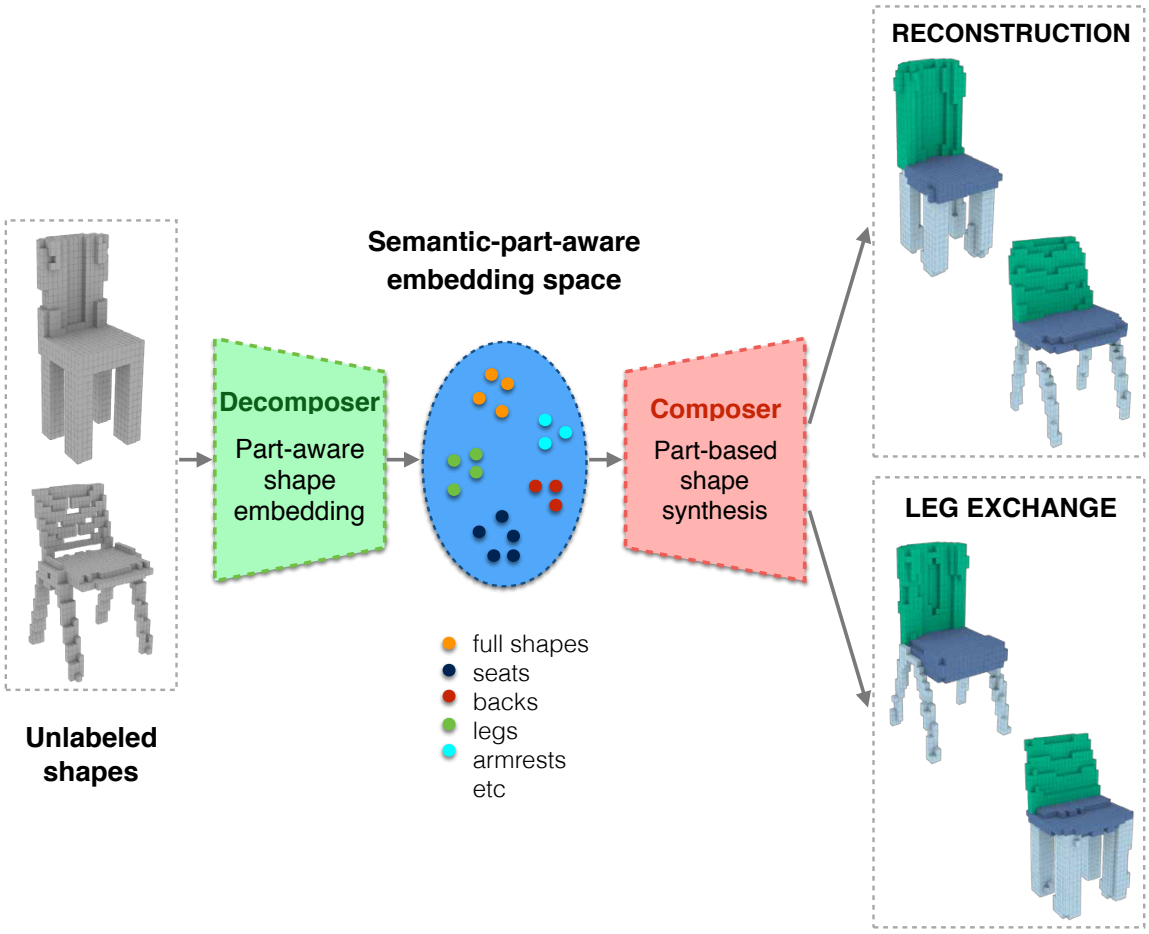


Figure 3.1: Given unlabeled shapes, the *Decomposer* maps them into a factorized latent space. The *Composer* can either reconstruct the shapes with semantic part labels, or create novel shapes, for instance, by exchanging chair legs.

3.2 Related Work

Learning-based shape synthesis Learning-based methods have been used for automatic synthesis of shapes from complex real-world domains; In a seminal work [119], *Kalogerakis et al.* used a probabilistic model, which learned both continuous geometric features and discrete component structure, for component-based shape synthesis and novel shape generation. The development of deep neural networks enabled learning high-dimensional features more easily; 3DGAN [290] uses 3D decoders and a GAN to generate voxelized shapes. Apart from generating shapes using a latent representation, some methods generate shapes from a latent representation with *structure*. S^2-GAN [284] generate the shape and texture for a 3D scene in a 2-stage manner. GRASS [156] generate

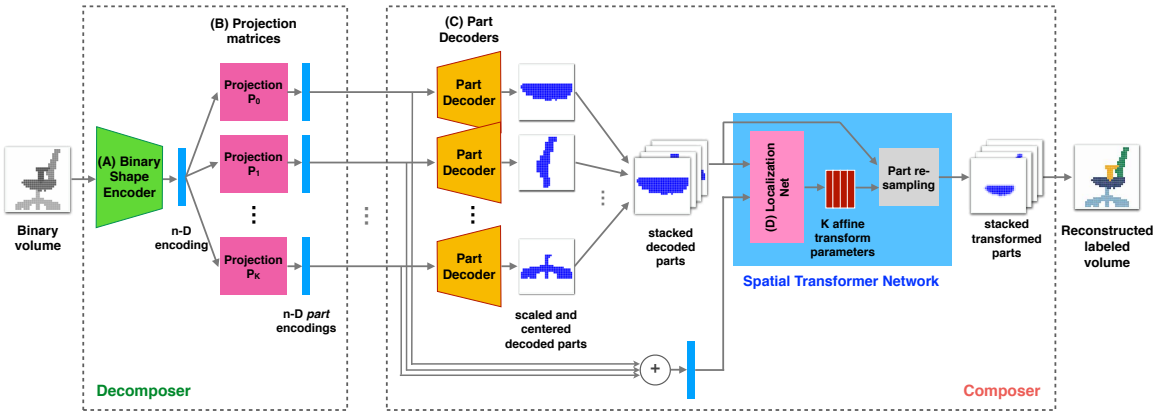


Figure 3.2: Our instantiation of a *Decomposer-Composer* architecture which enables explicit part-aware (deep-learning-based) modeling of shapes.

shapes in two stages: first, by generating orientated bounding boxes, and then a detailed geometry within those bounding boxes. Nash and Williams [196] use use point cloud shape representation and a VAE to learn a probabilistic latent space of shapes; however, they require all training data to be in point-to-point correspondence. In a related work [280], *Wang et al.* introduced a 3D GAN-based generative model for 3D shapes, which produced segmented and labeled into parts shapes. Unlike the latter approach, our network does not use predefined subspaces for part embedding, but learns to project the latent code of the entire shape to the subspaces corresponding to codes of different parts.

In concurrent efforts, several deep architectures for part based shape synthesis were proposed [242, 155, 292, 191]. *Schor et al* [242] utilized point-base shape representation, while operating on input models with known per-point parts labels. *Li et al.* [155] and [292] proposed two generative networks for part-based shape synthesis, operating on labeled voxelized shapes. *Mo et al.* [191] introduced a hierarchical graph network for learning structure-aware shape generation.

Spatial transformer networks Spatial transformer networks (STN) [115] allow to easily incorporate deformations into a learning pipeline. Kurenkov et al [144] retrieve a 3D model from one RGB image and generate a deformation field to modify it. Kanazawa et al [120] model articulated or soft objects with a template shape and deformations. Lin et al [159] use STNs iteratively, to warp a foreground onto a background, and use a GAN to constrain the composition results to the natural image manifold. *Hu et al.* [108] use a 3D STN to scale and translate objects given as volumetric grids, as a part of scene generation network. Inspired by this line of work, we incorporate an affine transformation module into our network. This way, the generation module only needs to generate normalized parts, and the deformation module transforms and assembles the parts together.

Deep latent space factorization Several approaches suggested to learn disentangled latent spaces for image representation and manipulation. β-VAE [104] introduce an adjustable hyperparameter β that balances latent channel capacity and independence constraints with reconstruction accuracy. InfoGAN [52] achieves the disentangling of factors by maximizing the mutual information between certain channels of latent code and image labels. Some approaches disentangle the image generation process using intrinsic decomposition, such as albedo and shading [250], or normalized shape and deformation grid [216, 249]. The proposed approach differs from [216, 249, 250] in that it maps both full and partial shapes into the same low dimensional embedding space, while in [216, 249, 250], different components have their own separated embedding spaces.

Projection in neural networks Projection is widely used in representation learning. It can be used for transformation from one domain to another domain [26, 213, 214], which is useful for tasks like translation in natural language processing. For example, Senel et al [244] use projections to map word vectors into semantic categories. In this work, we use a projection layer to transform a whole shape embedding into semantic part embeddings.

3.3 The Decomposer-Composer Model

Decomposer network

The Decomposer network is trained to embed unlabeled shapes into a factorized embedding space, reflecting the shared semantic structure of the shape collection. To allow for composite shape synthesis, the embedding space has to satisfy the following two properties: factorization consistency across input shapes, and existence of a simple shape composition operator to combine latent representations of different semantic factors. We propose to model this embedding space V as a *direct sum of subspaces* $\{V_i\}_{i=1}^{K}$, where K is the number of semantic parts, and each subspace $\{V_i\}$ corresponds to a semantic part i, thus satisfying the factorization consistency property. The second property is ensured by the fact that every vector $v \in V$ is given by a sum of unique $v_i \in V_i$ such that $V = V_1 \oplus ... \oplus V_k$, and part composition may be performed by part embedding summation. This also implies that the decomposition and composition operations in the embedding space are fully reversible.

A simple approach for such factorization is to split the dimensions of the n-dimensional embedding space into K coordinate groups, each group representing a certain semantic part-embedding. In this case, the full shape embedding is a concatenation of part embeddings, an approach explored

in [280]. This, however, puts a hard constraint on the dimensionality of part embeddings, and thus also on the representation capacity of each part embedding subspace. Given that different semantic parts may have different geometric complexities, this factorization may be sub-optimal.

Instead, we perform a data-driven learned factorization of the embedding space into semantic subspaces. We use *learned* part-specific projection matrices, denoted by $\{P_i\}_{i=1}^{K} \in \mathbb{R}^{n \times n}$. To ensure that the aforementioned two factorization properties hold, the projection matrices must form a *partition of the identity* and satisfy the following three properties

$$\begin{aligned} &(1)\ \ P_i^2 = P_i, \forall i, \\ &(2)\ \ P_i P_j = 0 \text{ whenever } i \neq j, \\ &(3)\ \ P_1 + ... + P_K = I, \end{aligned} \tag{3.1}$$

where 0 and I are the all-zero and the identity matrices of size $n \times n$, respectively.

In practice, we efficiently implement the projection operators using fully connected layers without added biases, with a total of $K * n^2$ variables, constrained as in Equation 3.1. The projection layers receive as input a whole shape encoding, which is produced by a 3D convolutional shape encoder. The parameters of the shape encoder and the projection layers are learned simultaneously. The resulting architecture of the Decomposer network is schematically described in Figure 3.2, and a detailed description of the shape encoder and the projection layer architecture is given in the Appendix B.

Composer network

The composer network is trained to reconstruct shapes with semantic part labels from *sets* of semantic part embedding coordinates. The simplest composer implementation would consist of a single decoder mirroring the whole binary shape encoder (see Figure 3.2), producing a semantically labelled reconstructed output shape. Such approach was used in [280], for instance. However, this straightforward method is known to fail in reconstructing thin volumetric shape parts and other fine shape details. To address this issue, we use a different approach, where we first separately reconstruct scaled and centered shape parts, using a *shared part decoder*. We then produce *per-part transformation parameters* and use them to deform the parts in a coherent manner, to obtain a complete reconstructed shape.

In our model, we make the simplifying assumption that it is possible to combine a given set of parts into a plausible shape by transforming them with per-part affine transformations and translations.

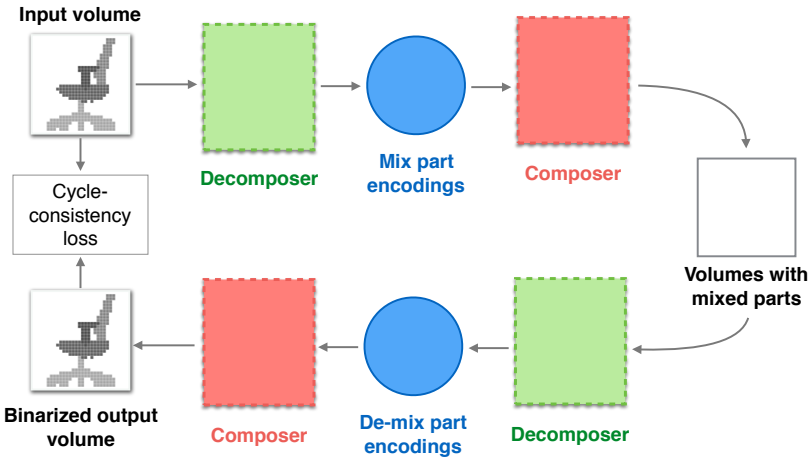


Figure 3.3: Schematic description of the cycle consistency constraint. See Section 3.3 for details.

While the true set of transformations which produce plausible shapes is significantly larger and more complex, our experiments demonstrate that the proposed simplified model is successful at producing geometrically and visually plausible results. This in-network part transformation is implemented using a 3D spatial transformer network (STN) [115]. It consists of a localization net, which produces a set of 12-dimensional affine transformations (including translations) for all parts, and a re-sampling unit, which transforms and places the reconstructed part volumes at their correct locations in the full shape. The SNT receives as input both the reconstructed parts from the part decoder, and the sum of part encodings, for best reconstruction results. The resulting Composer architecture is schematically described in Figure 3.2; its detailed description is given in the Appendix B.

We note that the proposed approach is related to the two-stage shape synthesis approach of [156], in which a GAN is first used to synthesize oriented bounding boxes for different parts, and then the part geometry is created per bounding box using a separate part decoder. Our approach is similar, yet it works in a reversed order. Namely, we first reconstruct part geometry, and then compute per-part affine transformation parameters, which are a 12-dimensional equivalent of the oriented part bounding boxes in [156]. Similarly to [156], this two stage approach improves the reconstruction of fine geometric details. However, unlike [156], where the GAN and the part decoder where trained separately, in our approach the two stages belong to the same reconstruction pipeline, trained simultaneously and end-to-end.

Cycle consistency

Our training set is comprised of 3D shapes with ground-truth semantic part-decomposition; It does not include any training examples of synthesized composite shapes. Existing methods for such shape assembly task operate on 3D meshes with very precise segmentations, and often with additional knowledge about part connectivity [300, 246, 119]. These methods cannot be applied to a dataset like ours, to produce a sufficiently large set of plausible new shapes (constructed from existing parts) to use for training a deep network for composite shape modelling. In order to circumvent this difficulty, and train the net to produce non-trivial part transformations for geometrically and semantically plausible part arrangements, we use a *cycle consistency* constraint. It has been previously utilized in geometry processing [197], image segmentation [279], and more recently in neural image transformation [216, 323].

Specifically, given a batch of M training shapes $\{X\}_{i=1}^{M}$, we map them to the factored latent space using the Decomposer, producing K semantic part encodings per input shape. We the randomly mix the part encodings of the shapes in the batch, while ensuring that after the mixing each of the new M encoding sets includes exactly one embedding coordinate per semantic part. We then reconstruct the shapes with correspondingly mixed parts using the Composer. After that, these new shapes are passed to the Decomposer-Composer pipeline once again, while de-mixing part encodings produced by the second Decomposer application, to re-store the original encoding-to-shape association. The cycle consistency requirement means that the final shapes are as similar as possible to the original M training shapes. We enforce it using the cycle consistency loss described in the next section. The double application of the proposed network with part encoding mixing and de-mixing is schematically described in Figure 3.3.

Loss function

Our loss function is defined as the following weighted sum of several loss terms

$$L = w_{\text{PI}}\mathcal{L}_{\text{PI}} + w_{\text{part}}\mathcal{L}_{\text{part}} + w_{\text{trans}}\mathcal{L}_{\text{trans}} + w_{\text{cycle}}\mathcal{L}_{\text{cycle}}. \tag{3.2}$$

The weights compensate for the different scales of the loss terms, and reflect their relative importance.

Partition of the identity loss $\mathcal{L}_{\text{PI}}$ measures the deviation of the predicted projection matrices from the optimal projections, as given by Equation 3.1.

$$\mathcal{L}_{\text{proj}}(P_1, ..., P_k) = \sum_{i=1}^{K} \|P_i^2 - P_i\|_F^2 + \sum_{\substack{i,j=1,\\ i \neq j}}^{K} \|P_i P_j\|_F^2 + \|P_1 + ...P_K - I\|_F^2. \tag{3.3}$$

Part reconstruction loss $\mathcal{L}_{\text{part}}$ is the binary cross-entropy loss between the reconstructed centered and scaled part volumes and their respective ground truth part indicator volumes, summed over K parts.

Transformation parameter loss $\mathcal{L}_{\text{trans}}$ is an $L2$ regression loss between the predicted and the ground truth 12-dimensional transformation parameter vectors, summed over K parts. Unlike in the original STN approach [115], we found that direct supervision over the transformation parameters is critical for our network convergence.

Cycle consistency loss $\mathcal{L}_{\text{cycle}}$ is a binary cross-entropy loss between ground truth input volumes and their reconstructions, obtained using two applications of the proposed network, as described in Section 3.3.

Training details

The network was implemented in TensorFlow [1], and trained for 500 epochs with batch size 32. We used Adam optimizer [129] with learning rate 0.0001, decay rate of 0.8, and decay step size of 40 epochs. We found it was essential to first pre-train the binary shape encoder, projection layer and part decoder parameters separately for 150 epochs, by minimizing the part reconstruction and the partition of the identity losses and using $w_{\text{trans}} = w_{\text{cycle}} \approx 0$, for improved part reconstruction results. We then train the parameters of the spatial transformer network for another 100 epochs, while keeping the rest of the parameters fixed. After that we resume the training with all parameters and the cycle consistency loss to fine-tune the network parameters. The optimal loss combination weights were empirically detected using the validation set, and set to be $w_{\text{PI}} = 0.1, w_{\text{part}} = 100, w_{\text{trans}} = 0.1, w_{\text{cycle}} = 0.1$. The network was trained on each shape category separately.

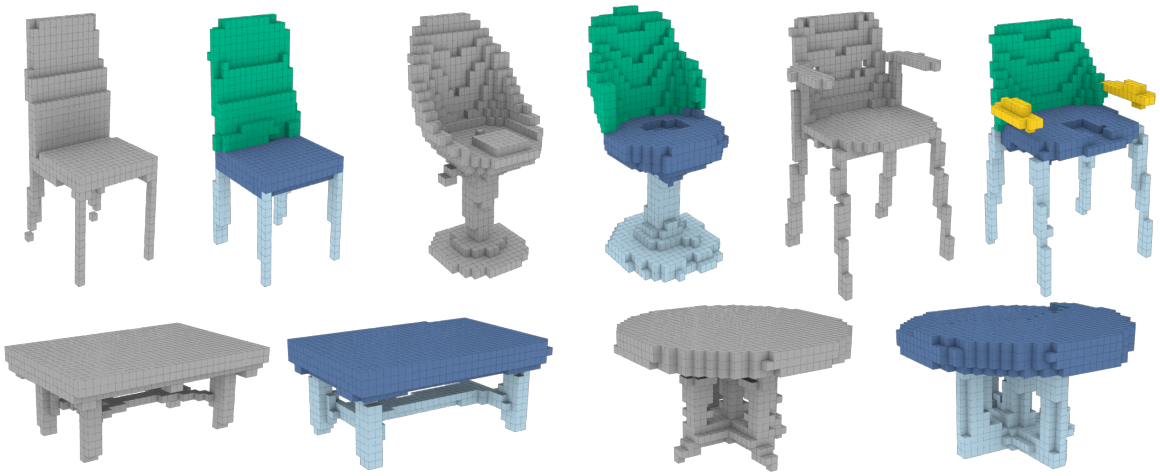


Figure 3.4: Reconstruction results of the proposed pipeline, for chair and table shapes. Gray shapes are the input test shapes; the results are colored according to the part label.

3.4 Experiments

Dataset In our experiments, we used the models from the ShapeNet 3D data collection [46], with part annotations produced by *Yi et al.* [307]. The shapes were converted to $32 \times 32 \times 32$ occupancy grids using binvox [199]. Semantic part labels were first assigned to the occupied voxels according to the proximity to the labeled 3D points, and the final voxel labels were obtained using graph-cuts in the voxel domain [38]. We used the official ShapeNet train, validation and test data splits in all our experiments. Additional results for $64 \times 64 \times 64$ occupancy grids can be found in the Appendix B.

Shape reconstruction

Figure 3.4 presents the results of reconstructing semantically labeled shapes from unlabelled input shapes, using the proposed network. Note that since our method performs separate part reconstruction with part decoders and part placement with an STN, it may produce less accurate part reconstruction, as compared to segmentation approaches - for example, the handles of the reconstructed rightmost chair in Figure 3.4. But, as illustrated by our quantitative study in Section 3.4, this allows us to perform better part-based shape manipulation.

Composite shape synthesis

Shape composition by part exchange In this experiment, we used our structured latent space to randomly swap corresponding embedding coordinates of pairs of input shapes (e.g.,, embedding

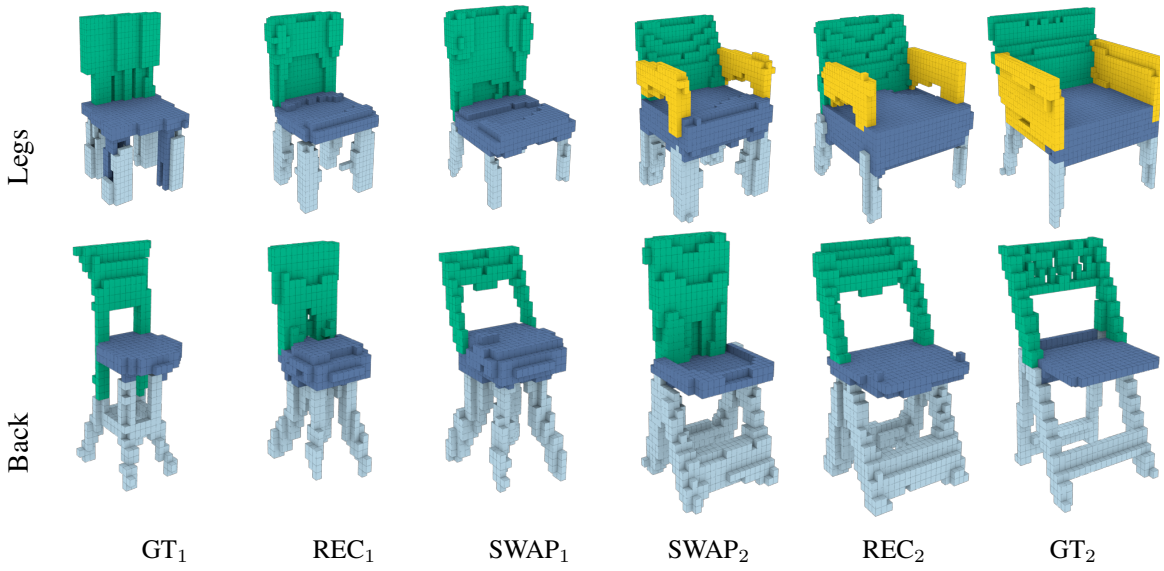


Figure 3.5: Single part exchange experiment. $GT_{1/2}$ denote ground truth shapes, $REC_{1/2}$ - reconstruction results, $SWAP_{1/2}$ - part exchange results. Unlabeled shapes were used as an input.

coordinates of legs or seats of two chairs), and reconstruct the new shapes using the Composer. The results are shown in Figure 3.5, and demonstrate the ability of our system to perform accurate part exchange, while deforming the geometry of both the new and the existing parts to obtain a plausible result.

Shape composition by random part assembly In this experiment we tested the ability of the proposed network to assemble shapes from random parts using our factorized latent space. Specifically, we mapped batches of input shapes into the latent space using the Decomposer, and created new shapes by randomly mixing the part embedding coordinates of the shapes in the batch, and reconstructing new shapes using the Composer. The results are shown in Figure 3.6, for chairs and tables, and illustrate the ability of the proposed method to combine parts from different shapes, scale and translate them so that the resulting shape looks realistic.

Full and partial interpolation in the embedding space In this experiment, we tested reconstruction from linearly interpolated embedding coordinates of complete shapes, as well as of a single semantic part. For the latter, we performed the part exchange experiment, described above, and interpolated the coordinates of that part, while keeping the rest of part embedding coordinates fixed. The results are shown in Figure 3.7.

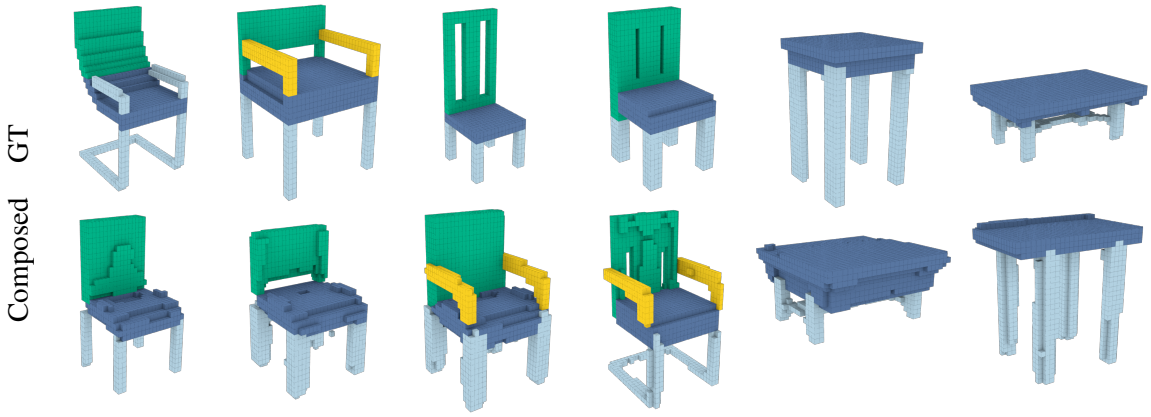


Figure 3.6: Shape composition by random part assembly. The top row shows the ground truth (GT) shapes, and the bottom row - shapes assembled using the proposed approach (see Section 3.4). Unlabeled shapes were used as an input.

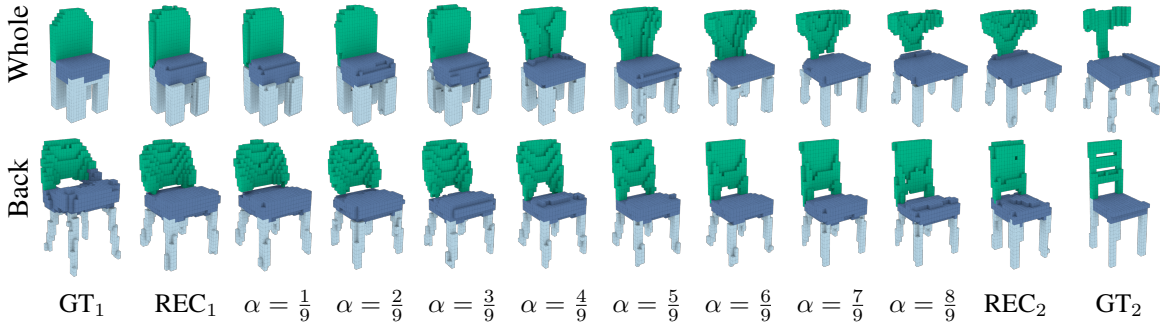


Figure 3.7: Example of a whole (top) and partial (bottom) shape interpolation. $GT_{1/2}$ denote original models, $REC_{1/2}$ - their reconstructions, and linear interpolation results are in the middle. Unlabeled shapes were used as an input.

Latent space and projection matrix analysis

The latent space obtained using the proposed method exhibits good separation into subspaces corresponding to different semantic parts (see Figure 3.8). The projection matrices, while not not being strictly orthogonal, as required for the partition of the identity (3.1), have low effective ranks, which is in line with the clear separation into non-overlapping subspaces produced by them. See the Appendix B for a visualization the projection matrices ranks.

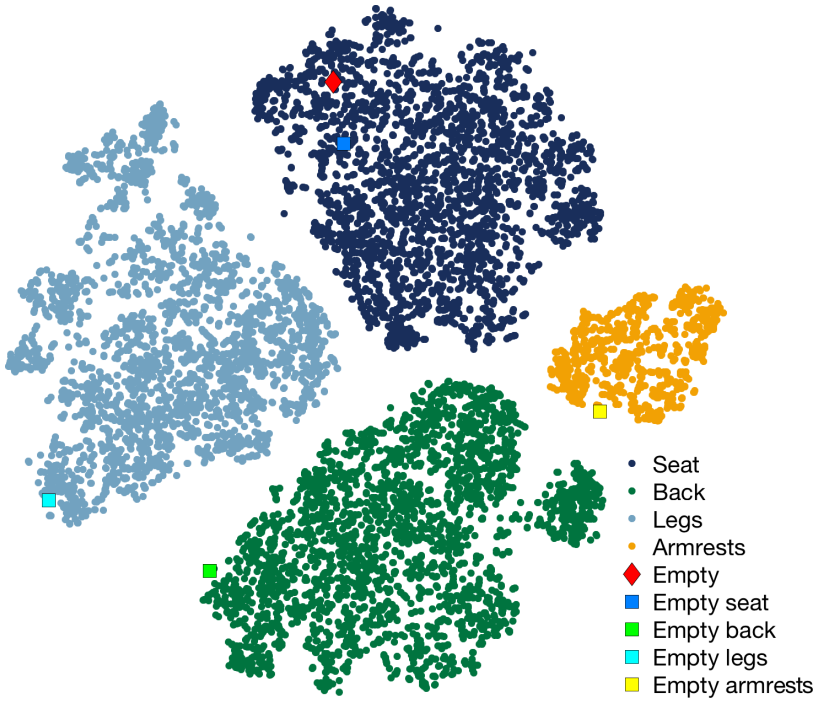


Figure 3.8: T-SNE visualization [174] of the produced embedding space, using both train and test shape embedding coordinates. The "empty" part coordinates correspond to the embedding produced for non-existing parts.

Ablation study and comparison with existing approaches

Ablation study

To highlight the importance of the different elements of our approach, we conducted an ablation study, where we used several variants of the proposed method, listed below.

Fixed projection matrices Instead of using learned projection matrices in the Decomposer, the n-dimensional shape encoding is split into K consecutive equal-sized segments, which correspond to different part embedding subspaces. This is equivalent to using constant projection matrices, where the elements of the rows corresponding to a particular embedding space dimensions are 1, and the rest of the elements are 0.

Composer without STN We substituted the proposed composer, consisting of the part decoder and the STN, with a single decoder producing a labeled shape. The decoder receives the sum of part encodings as an input, processes it with two FC layers to combine information from different parts, and then reconstructs a shape with parts labels using a series of deconvolution steps, similar to the

Method \ Metric	mIoU	mIoU (parts)	Connectivity			Classifier accuracy		
	Rec.	Rec.	Rec.	Swap	Mix	Rec.	Swap	Mix
Our method	0.64	0.65	**0.82**	**0.71**	**0.65**	**0.95**	0.89	**0.83**
W/o cycle loss	0.63	0.66	0.74	0.62	0.54	0.93	0.84	0.80
Fixed projection	0.63	0.65	0.72	0.61	0.58	0.94	0.86	0.77
Composer w/o STN	**0.75**	**0.8**	0.69	0.48	0.23	**0.95**	**0.9**	0.71
Naive placement	-	-	-	0.68	0.62	0.61	0.47	0.21
ComplementMe	-	-	-	**0.71**	0.47	-	0.66	0.43
Segmentation+STN	-	-	-	0.41	0.64	-	0.64	0.36

Table 3.1: Ablation study results. The evaluation metrics are mean Intersection over Union (*mIoU*), per-part mean IoU (*mIoU (parts)*), shape *connectivity* measure, and binary shape *classifier accuracy*. Rec., Swap and Mix stand for the shape reconstruction, part exchange and random part assembly experiment results, respectively (see Section 3.4 for a detailed description of the compared methods and the evaluation metrics.)

part decoder in the proposed architecture.

Without cycle loss We removed the cycle loss component during the network training.

Comparison with existing methods

Most existing methods for composite shape modeling operate on triangulated meshes with precise part segmentation. Hence, they are not directly applicable to the large-scale ShapeNet dataset with less precise segmentation, preventing a fair comparison. We therefore added the following comparisons with modern neural-net-based techniques: we combined the state-of-the-art ComplementMe method [263] with a 3D-CNN segmentation network [219]. From the former we used the *component placement network*, which, given a partial shape and a complementary component, produces a 3-D translation to place the component correctly w.r.t. the partial shape. To produce the "to-be-added" component we used a 3D-CNN segmentation network, described in [219], which achieved a state-of-the-art mean Intersection over Union (mIoU) of 0.91 on the test set. Together, these two networks replace our proposed Decomposer-Composer. Both networks were trained using the same training data as the proposed method. This method is denoted by *ComplementMe* in Table 3.1.

For an additional comparison, instead of the placement network of ComplementMe we utilized the spatial transformer network. Here, the STN was trained using the ground truth shape parts, and at test time it was applied to the results of the segmentation network, described above. This method is denoted by *Segmentation+STN* in Table 3.1.

Finally, we compared the proposed method to a baseline shape composition network. Given ground-truth shape parts, it composes new shapes from these parts by placing them at their original locations in the source shapes they were extracted from. All the shapes in our dataset are centered and uniformly scaled to fill the unit volume, and there exist clusters of geometrically and semantically similar shapes. Thus, we can expect that even this naive approach without part transformations will produce plausible results in some cases. This method is denoted by *Naive placement* in Table 3.1.

Evaluation metrics

Mean Intersection over Union (mIoU) is commonly used to evaluate the performance of segmentation algorithms [164]. Here, we use it as a metric for the reconstruction quality. We computed the mIoU for both actual-sized reconstructed parts, and scaled and centered parts (when applicable). We denote the two measures by *mIoU* and *mIoU (parts)* in Table 3.1.

Connectivity In part based shape synthesis, one pathological issue is that parts are often disconnected, or penetrate each other. Here, we would like to benchmark the quality of part placement, in terms of part connectivity. For each $32 \times 32 \times 32$ volume, we compute the frequency of the shape forming a single connected component, and report it as *Connectivity* in Table 3.1.

Classification accuracy To measure the shape composition quality of different methods, we trained a binary neural classifier to distinguish between ground-truth whole chairs (acting as positive examples) and chairs produced by naively placing random chair parts together (acting as negative examples). To construct the negative examples, we randomly combined ground-truth shape parts, by adding a certain semantic part only once, and placing the parts at their original locations in the source shapes they were extracted from. In addition, we removed negative examples assembled from parts from geometrically and semantically similar chairs, since such part arrangement could produce plausible shapes incorrectly placed in the negative example set. The attained classification accuracy on the test set was $\sim 88\%$. For a given set of chairs, we report the average classification score. Details of the network can be found in the Appendix B. The results are reported as *Classifier accuracy* in Table 3.1.

Symmetry The chair shapes in the ShapeNet are predominantly bilaterally symmetric, with vertical symmetry plane. Thus, similar to [280], we evaluated the symmetry of the reconstructed shapes, and defined the *Symmetry score* as the percentage of the matched voxels (filled or empty) in the reconstructed volume and its reflection with respect to the vertical symmetry plane. We performed this evaluation using binarized reconstruction results, effectively measuring the global symmetry of

the shapes. For the evaluation, we used the shapes in the test set (690 shapes), and conducted three types of experiments: shape reconstruction, single random part exchange between a pair of random shapes, shape composition by random part assembly.

Evaluation result discussion

According to all metrics, our method outperforms or performs on par with all the baselines, and *significantly* outperforms other existing methods. This shows that our design choices - the cycle loss, learned projection matrices and usage of the STN, help to achieve plausible results both when reconstructing shapes, and when performing composite shape synthesis. This is especially pronounced in the connectivity test results, illustrating that these design choices are necessary for achieving good assembly quality.

In the classifier accuracy test and the symmetry test the proposed method performs slightly better or on par with all baselines considered in the ablation study. It seems that both these tests are less sensitive to disconnected shape components, and most advantage that the proposed method achieves over the baselines is in its composition robustness. As expected, the naive placement also achieves high symmetry score, since it preserves the symmetry of the ground-truth parts during shape assembly.

According to the mIoU and per-part mIoU metrics, the proposed method performs on par with all baselines, except when using the simple version of the Composer, without STN. This follows from the fact that the proposed system, while reconstructing better fine geometry features, decomposes the problem into two inference problems, for the geometry and the transformation, and thus does not produce as faithful reconstruction of the original model as the simple decoder. Notably, this version of the architecture achieves worst connectivity scores for all compared methods, which follows from the fact that such a Decomposer is unable to faithfully reconstruct fine shape details.

3.5 Conclusion

This chapter presented a novel neural architecture termed Decomposer-Composer capable of performing structure-aware 3D shape modelling. Specifically, our network is able to generate a factorized latent shape representation, where different semantic part embedding coordinates lie in separate linear subspaces. The subspace factorization allows us to perform shape manipulation via part embedding coordinates, exchange parts between shapes, or synthesize novel shapes by assembling a shape from random parts. Qualitative results show that the proposed system can generate high fidelity 3D shapes

and meaningful part manipulations. Quantitative results shows we are competitive in the mIOU, connectivity, symmetry and classification benchmarks.

While the proposed approach makes a step toward automatic shape-from-part assembly, it has several limitations. First, while we can generate relatively high-fidelity shapes at a low resolution, memory limitations do not allow us to work with voxelized shapes of higher resolution. Memory-efficient architectures, such as OctNet [229] and PointGrid [149], may help alleviate this constraint. Alternatively, using point-based shape representations and compatible deep network architectures, such as [219], may also reduce the memory requirements and increase the output resolution.

Secondly, we made a simplifying assumption that a plausible shape can be assembled from parts using per-part affine transformations, which represent only a subset of possible transformations. While this assumption simplifies the training, it is quite restrictive in terms of the deformations we can perform. Future work is needed, for incorporating more general transformations with higher degree of freedom, such as 3D thin plate splines or a general deformation fields. Finally, for this work we have been using a cross-entropy loss to measure a shape's reconstruction quality; it would be interesting to investigate the use of a GAN-based losses like those of Chapter 2, adapted in this structure-aware shape generation context.

Chapter 4

Building a Latent Space from 3D Shape Differences

4.1 Overview

The basic learning paradigm of the generative neural networks presented so far: the AutoEncoder of Chapter 2, and the part-aware AutoEncoder of Chapter 3, was based on decoding the shape of an object given a low-dimensional encoding of *itself*. In this chapter we will change gears by introducing a radically different idea. Instead of encoding directly a representation of the output we wish to decode, **we will create and learn to decode an operator that reflects how the input object shape is different from another object shape**. That is, we will make use of an indirect and implicit representation of a shape in the context of a collection. Namely, the "operator" we will learn to decode will capture how the input shape is different from a fixed "base" shape, e.g., it will capture how one can deform the input to match the base shape. Specifically, we will use the linear shape-difference operators introduced in the seminal paper of *Rustamov et al.* [239]; which assumes the (rich) supervision of **pairwise correspondences** among shapes and expresses their differences w.r.t. to the geometry of their underlying mesh manifold. Crucially, such linear shape-differences, can be compactly encoded as small-sized matrices making convolutional learning on them straightforward. Furthermore, as we show with the experiments presented in this chapter one can exploit the matrix nature of the proposed representation to apply **multipicative** linear algebra when doing latent-based shape-interpolations and analogies, with improved results compared to those of the linear approach presented in Chapter 3 (see Figures 4.1 and 4.8 for some examples).

It is worth pointing out, that this is the third and final generative network presented in this thesis,

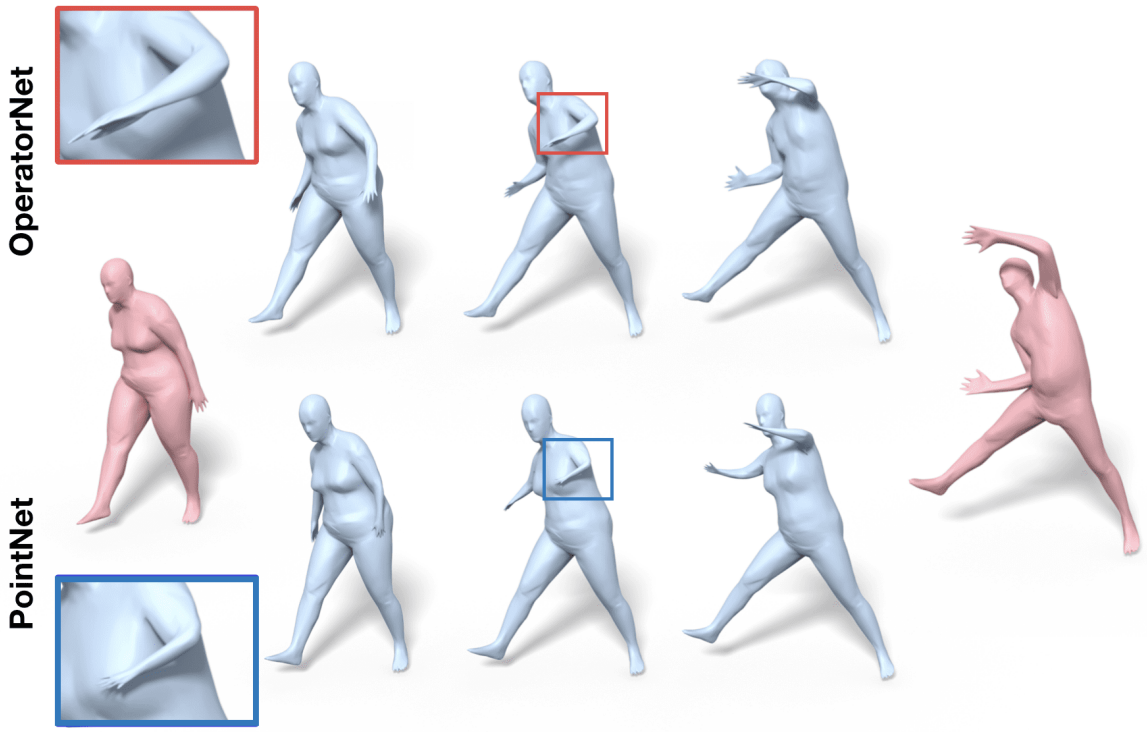


Figure 4.1: Shape interpolation via OperatorNet (top) and a PointNet-based AutoEncoder, similar to those presented in Chapter 3 (bottom).The interpolations based on the approach presented in the chapter (OperatorNet) are more smooth and less distorted.

and the described approach and included applications and experiments, (primarily) concern shapes represented as meshes. Moreover, while the approach taken in this chapter is purely geometrical and algorithmic, i.e., it examines how shapes are different based on correspondence-induced linear-algebraic constructs, it is highly related to the upcoming chapter. Namely, in Chapter 5 we also investigate geometric differences between shapes, but our focus there is how such differences are being conceived and expressed by humans in their "natural" way, i.e., in language. Last, it should be pointed out that the materials of this chapter are based on our work *OperatorNet: Recovering 3D Shapes from Difference Operators* [111]. Two closely related earlier works, where we also explored the analysis of shape collections based on correspondence-induced matrices, but without focusing on deep-learning, are our works [110, 109].

To summarize, our specific contributions in this domain are:

- We propose a learning-based pipeline to reconstruct 3D shapes from a set of difference operators.

- We propose a novel formulation of extrinsic shape difference, which complements the intrinsic operators formulated in [239].
- We demonstrate that by applying algebraic operations on shape differences, we can synthesize new operators and thus new shapes via OperatorNet, enabling shape manipulations such as interpolation and analogy.

4.2 Related Work

Shape reconstruction Some of the key ideas presented in this chapter are closely related to prior works related to shape reconstruction from intrinsic operators, which were recently considered in [35, 61]. In these works, several advanced and purely geometric optimization techniques were proposed that give satisfactory results in the presence of full information [35] or under strong (extrinsic) regularization [61]. These works have also laid the theoretical foundation for shape recovery by demonstrating that shape difference operators, in principle, contain complete information necessary for recovering the shape embedding (e.g., Propositions 2 and 4 in [61]). On the other hand, these methods also highlight the practical challenges raised by this approach to reconstruction: when reconstructing a shape without any knowledge of the collection or "shape space" that it belongs to. In contrast, in this chapter we show that when using shape difference representations in a learning-based framework, realistic 3D shapes can be recovered efficiently, and moreover that entirely new shapes can be synthesized using the algebraic structure of difference operators, e.g., for shape interpolation and analogy.

Shape representations for learning The materials of this chapter are also related to the several recent techniques aimed at applying deep learning methods to shape analysis. As discussed in Section 1.2 one of the main challenges deep learning faces when dealing with 3D shapes is defining a meaningful notion of convolution, while ensuring invariance to basic 3D transformations. Despite the tremendous progress in the last few years in designing methods that attempt to address these bottlenecks [253, 181, 282, 41, 179], defining a shape representation that is compact, lends itself naturally to learning, while being invariant to the desired class of transformations (e.g., rigid motions) and not limited to a particular topology, remains a challenge. As we show below, the proposed representations used in this chapter are well-suited for learning applications, and especially for encoding and recovering geometric structure information.

Shape space Exploring the structure of shape spaces has been an attractive research topic for a long time. Classical PCA-based models, e.g., [17, 97], and more recent shape space models, adapted to specific shape classes such as humans or animals, such as [167, 326], or parametric model collections [243]; all typically leverage the fact that the space of "realistic" shapes is significantly smaller than the space of all possible embeddings. This has also recently been exploited in the context of learning-based shape synthesis applications for shape completion [160], interpolation [28] and point cloud reconstruction like those presented in Chapter 2. These techniques heavily leverage the recent proliferation of large shape collections such as DFAUST [34] and ShapeNet [46] to name a few. At the same time, it is not clear if, for example, the commonly used *linear interpolation* of latent vectors is well-justified, leading to unrealistic synthesized shapes. Instead, the shape difference operators that we use in this chapter satisfy a well-founded multiplicative algebra, which, as we show, can naturally create more realistic synthetic interpolations.

4.2.1 Preliminaries & notations

Discretization of Shapes Throughout this Section, we assume that a shape is given as a triangular mesh $(\mathcal{V}, \mathcal{T})$, where $\mathcal{V} = \{v_1, v_2, \cdots, v_n\}$ is the vertex set, and $\mathcal{T} = \{(v_i, v_j, v_k) | v_i, v_j, v_k \in \mathcal{V}\}$ is the set of triangles encoding the connectivity of the vertices.

Laplace-Beltrami Operator We associate with each shape a discretized Laplace-Beltrami operator, $\mathcal{L} \coloneqq A^{-1}W$, using the cotangent weight scheme from [183, 210], where W is the cotangent weight (stiffness) matrix, and A is the diagonal lumped area (mass) matrix. Furthermore, we denote by $\mathbf{\Lambda}$ the diagonal matrix containing the k smallest eigenvalues, and by $\mathbf{\Phi}$ the corresponding eigenvectors of S, such that $W\mathbf{\Phi} = A\mathbf{\Phi}\mathbf{\Lambda}$. In particular, the eigenvalues stored in Λ are non-negative and can be ordered as $0 = \lambda_1 \leq \lambda_2 \leq \cdots$. The columns of Φ are sorted accordingly, and are orthonormal with respect to the area matrix, i.e. $\mathbf{\Phi}^T A\mathbf{\Phi} = \mathbf{I}_{k\times k}$, the $k \times k$ identity matrix. It is well-known that the Laplace-Beltrami eigenbasis provides a multi-scale understanding of a shape [151], and allows to approximate the space of functions via a subspace spanned by the first few eigenvectors of $\mathbf{\Phi}$.

Functional Maps The functional map framework was introduced in [201] primarily as an alternative representation of maps across shapes. In our context, given two shapes S_0, S_1 and a point-wise map T from S_1 to S_0, we can express the functional map $\mathbf{C}_{01}$ from S_0 to S_1, as follows:

$$\mathbf{C}_{01} = \mathbf{\Phi}_1^T A_1 \Pi_{01} \mathbf{\Phi}_0. \tag{4.1}$$

Here, A_1 is the area matrix of S_1, and Π_{01} is a binary matrix satisfying $\Pi_{01}(p,q) = 1$ if $T(p) = q$ and 0 otherwise. Note that $\mathbf{C}_{01}$ is a $k_1 \times k_0$ matrix, where k_1, k_0 is the number of basis functions chosen on S_1 and S_0. This matrix allows to transport functions as follows: if f is a function on S_0 expressed as a vector of coefficients $\mathbf{f}$, s.t. $f = \mathbf{\Phi}_0\mathbf{f}$, then $C_{01}\mathbf{f}$ is the vector of coefficients of the corresponding function on S_1, expressed in the basis of $\mathbf{\Phi}_1$.

In general, not every functional map matrix arises from a point-wise map, and might include for example soft correspondences, which map a point to a probability density function. All of the tools that we develop below can accommodate such general maps. This is a key advantage of our approach, as it does not rely on all shapes having the same number of points, and only requires the knowledge of functional map matrices, which can be computed using existing techniques [202, 161].

Intrinsic Shape Difference Operators Finally, to represent shapes themselves, we use the notion of shape difference operators proposed in [239]. Within our setting, they can be summarized as follows: given a base shape S_0, an arbitrary shape S_i and a functional map $\mathbf{C}_{0i}$ between them, let $\mathbf{K}_0$ (resp. $\mathbf{K}_i$) be a positive semi-definite matrix, which defines some inner product for functions on S_0 (resp. S_i) expressed in the corresponding bases. Thus, for a pair of functions f, g on S_0 expressed as vectors of coefficients $\mathbf{a}, \mathbf{b}$, we have $< f, g >= \mathbf{a}^T\mathbf{K}_0\mathbf{b}$.

Note that these two inner products $\mathbf{K}_0, \mathbf{K}_i$ are not comparable, since they are expressed in different bases. Fortunately, the functional map $\mathbf{C}_{0i}$ plays a role of basis synchronizer. Thus, a shape difference operator, which captures the difference between S_0 and S_i is given simply as:

$$\mathbf{D}_{0i}^K = \mathbf{K}_0^+(\mathbf{C}_{0i}^T\mathbf{K}_i\mathbf{C}_{0i}), \tag{4.2}$$

where $^+$ is the Moore-Penrose pseudo-inverse.

The original work [239] considered two intrinsic inner products, which using the notation above, can be expressed as: $\mathbf{K}^{L^2} = \mathbf{Id}$, and $\mathbf{K}^{H^1} = \mathbf{\Lambda}$. These inner products, in turn lead to the following shape differences operators, capturing area and angle distortions respectively:

$$\text{Area-based } (L^2): \qquad \mathbf{D}_{0i}^A = \mathbf{C}_{0i}^T\mathbf{C}_{0i}, \tag{4.3}$$

$$\text{Conformal } (H^1): \qquad \mathbf{D}_{0i}^C = \Lambda_0^+\mathbf{C}_{0i}^T\Lambda_i\mathbf{C}_{0i}. \tag{4.4}$$

These shape difference operators have several key properties. First, they allow to represent an arbitrary shape S_i, as a pair of matrices of size $k_0 \times k_0$, independent of the number of points, by requiring only a functional map between the base shape S_0 and S_i. Thus, the size of this

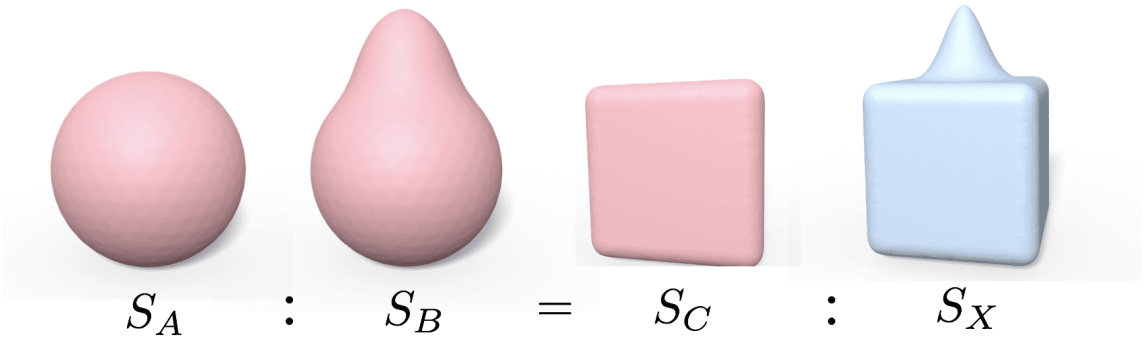


Figure 4.2: Illustration of shape analogy.

representation can be controlled by choosing an appropriate value of k_0 which allows to gain multi-scale information about the geometry of S_i, from the point of view of S_0. Second, and perhaps more importantly, these matrices are invariant to rigid (and indeed any intrinsic isometry) transformation of S_0 or S_i. Finally, previous works [61] have shown that shape differences *in principle* contain complete information about the intrinsic geometry of a shape. As we show below these properties naturally enable the use of learning applications for shape recovery.

Functoriality of Shape Differences Another useful property of the shape difference operators is *functoriality,* shown in [239], and which we exploit in our shape synthesis applications in Section 4.6. Given shape differences $\mathbf{D}_{0i}, \mathbf{D}_{0j}$ of shapes S_i and S_j with respect to a base shape S_0, functoriality allows to compute the difference $\mathbf{D}_{ij}$, without functional maps between S_i and S_j. Namely (see Prop. 4.2.4 in [60]):

$$\mathbf{D}_{ij} = \mathbf{C}_{0i}\mathbf{D}_{0i}^{+}\mathbf{D}_{0j}\mathbf{C}_{0i}^{-1} \,. \tag{4.5}$$

Intuitively, this means that shape differences naturally satisfy the *multiplicative algebra:* $\mathbf{D}_{0i}\mathbf{D}_{ij} = \mathbf{D}_{0j}$, up to a change of basis ensured by $\mathbf{C}_{0i}$.

This property can be used for *shape analogies*: given shapes S_A, S_B and S_C, to find S_X such that S_X relates to S_C in the same way as S_B relates to S_A (see the illustration in Figure 4.2), which can be solved by looking for a shape X that satisfies: $\mathbf{C}_{0C}^{+}\mathbf{D}_{CX}\mathbf{C}_{0C} = \mathbf{C}_{0A}^{+}\mathbf{D}_{AB}\mathbf{C}_{0A}$. In our application, we first create an appropriate $\mathbf{D}_{0X}$ and then use our network to synthesize the corresponding shape.

Finally, the multiplicative property also suggests a way of interpolation in the space of shape differences. Namely, rather than using basic linear interpolation between $\mathbf{D}_{0i}$ and $\mathbf{D}_{0j}$, we interpolate on the Lie algebra of the Lie group of shape differences, using the exponential map and its inverse,

which leads to:

$$\mathbf{D}(t) = \exp((1-t)\log(\mathbf{D}_{0i}) + t\log(\mathbf{D}_{0j})), t \in [0,1]. \tag{4.6}$$

Here $\exp$ and $\log$ are matrix exponential and logarithm respectively. Note that, around identity, the linearization provided by the Lie algebra is exact, and we have observed it to produce very accurate results in general.

4.3 Extrinsic Shape Difference

In theory, with purely intrinsic information one at the best can determine a shape up to isometric transformations, which in our setting means being able to capture the edge lengths of the mesh. Recovering the shape from its edge lengths, while possible in certain simple scenarios, nevertheless often leads to ambiguities, as highlighted in [61]. To alleviate such ambiguities, we propose to augment the existing purely intrinsic shape differences with a novel compatible shape difference operator that gives rise to a more complete characterization of shapes, and in turn boosts our reconstruction.

One basic approach that has been used to combine extrinsic or embedding-dependent information with the multi-scale Laplace-Beltrami basis, is simply to project the 3D coordinates, as functions onto the basis, to obtain three vectors of coefficients (one for each X, Y, Z coordinates): $\mathbf{f} = \mathbf{\Phi}^+ X$, where X is the $n_V \times 3$ matrix of vertex coordinates [151, 136]. Unfortunately representing a shape through $\mathbf{f}$, while also multi-scale and compact, has several limitations. First this representation is not rotationally invariant, and second, it does not provide information about intrinsic geometry, so that interpolation of coordinate vectors can easily lead to loss of shape area, for example.

Another option, which is more compatible with the shape difference representation and is rotationally invariant, is to encode the inner products of coordinate functions on each shape using the Gram matrix $G = XX^T$, where X is again the matrix of coordinate functions. Expressing G in the corresponding basis, and using Eq. (4.2) gives rise to a shape difference-like representation of the shape coordinates. Indeed, the following theorem (see proof in Appendix B.1.1) guarantees that the resulting shape difference representation contains the same information, up to rotational invariance, as simply projecting the coordinates onto the basis.

Theorem 1. *Let $\mathbf{G} = \mathbf{\Phi}^T A X X^T A \mathbf{\Phi}$ be the extrinsic inner product encoded in $\mathbf{\Phi}$, where A is the lumped mass matrix (as normalization factor), then one can recover the projections of the coordinate functions, X, on the subspace spanned by $\mathbf{\Phi}$ from $\mathbf{G}$ up to rigid transformations. In particular, when*

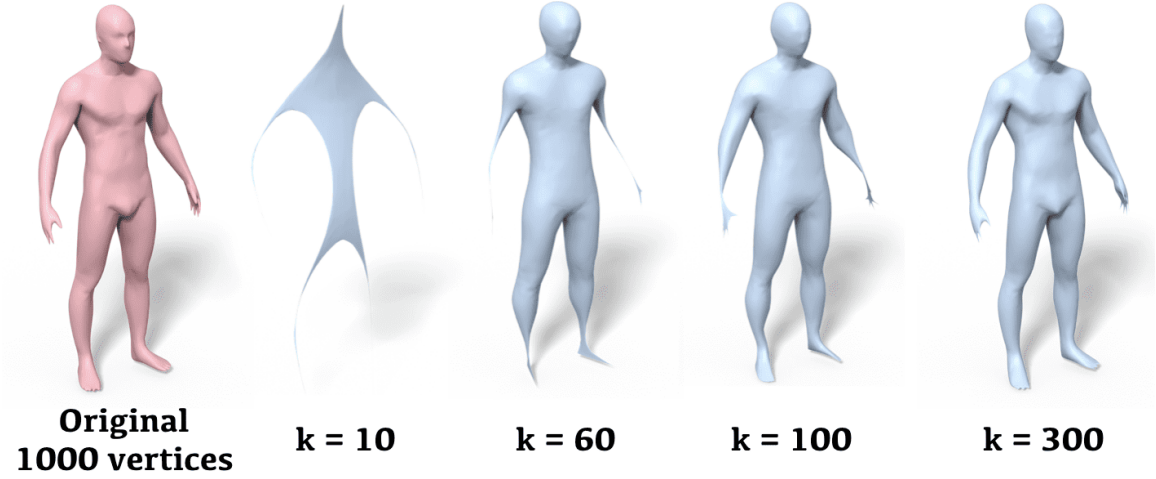


Figure 4.3: From left to right: original shape with 1000 vertices, the recovered embedding from $\mathbf{G}$ encoded in the leading k = 10, 60, 100 and 300 eigenbasis of the original shape.

$\mathbf{\Phi}$ *is a complete full basis, the recovery of* X *is exact.*

As an illustration of Theorem 1, we show in Figure 4.3 the embeddings recovered from $\mathbf{G}$ when the number of basis functions in $\mathbf{\Phi}$ range from 10 to 300.

Representing a shape via its Gram matrix $\mathbf{G}$ in either the full or the reduced basis has one key limitation, however, since the rank of $\mathbf{G}$ is at most 3, meaning that the majority of its eigenvalues are zero. This turns out to be an issue in applications, where gaining information about *the local* geometry of the shape is important, for example in our shape analogies experiments.

To compensate for this rank deficiency, we finalize our construction of the extrinsic inner product by making it Laplacian-like:

$$\mathbf{E}^D(i,j) = \begin{cases} -E(i,j) & \text{if } i \neq j, \\ \sum_{i \neq j} E(i,j) & \text{i = j.} \end{cases} \tag{4.7}$$

Where $E(i,j)$ is $\|v_i - v_j\|^2 A(i,i)A(j,j)$, i.e., the squared Euclidean distance between points v_i, v_j on the shape, weighted by the respective vertex area measure. Since $\mathbf{E}^D$ can be regarded as the Laplacian of a complete graph, all but one of its eigenvalues are strictly positive.

It is worth noting that the Gram matrix and the squared Euclidean distance matrix are inherently related and can be recovered from each other as is commonly done in the Multi-Dimensional Scaling literature [62].

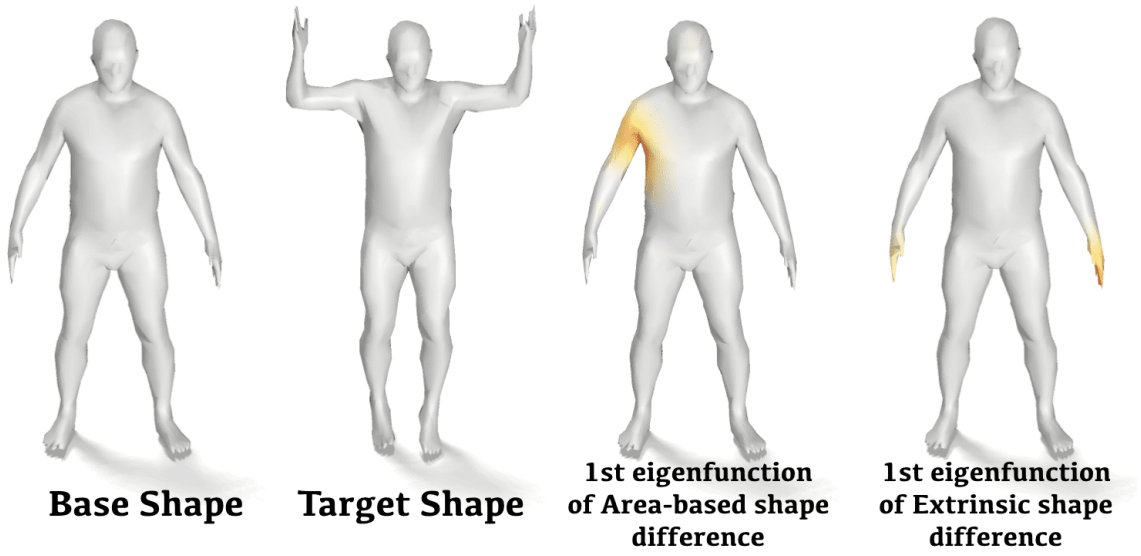


Figure 4.4: A pair of shapes are compared. The most area (resp. extrinsic) distorted region is captured by the leading eigenfunction of the area-based (resp. extrinsic) shape difference.

To summarize, given a base shape S_0, another shape S_i and a functional map $\mathbf{C}_{0i}$ we encode the extrinsic information of S_i from the point of view of S_0 as follows:

$$\mathbf{D}_i^E = (\mathbf{\Phi}_0^T \mathbf{E}_0^D \mathbf{\Phi}_0)^+ (\mathbf{C}_{0i}^T \mathbf{\Phi}_i^T \mathbf{E}_i^D \mathbf{\Phi}_i \mathbf{C}_{0i}). \tag{4.8}$$

In Figure 4.4, we compute $\mathbf{D}^A$ and $\mathbf{D}^E$ of the target shape with respect to the base, and color code their respective eigenfunctions associated with the largest eigenvalue, on the shapes to the right. As argued in [239] these functions capture the areas of highest distortion between the shapes, with respect to the corresponding inner products. Note that the eigenfunction of $\mathbf{D}^A$ captures the armpit where the local area changes significantly, while that of $\mathbf{D}^E$ captures the hand, where the *pose changes* are evident.

It is worth noting that in [61], the authors also propose a shape difference formulation for encoding extrinsic information, which is defined on the shape offset in order to extract information for surface normal. However, the construction of offset can lead to instabilities, and moreover, this formulation only gives information about local distances, making it hard to recover large changes in pose.

4.4 Network Details

Problem Setup Our general goal is to develop a neural network capable of recovering the coordinates functions of a shape, given its representation as a set of shape difference operators. This way we aim to solve the same problem considered in [35, 61]. However, unlike these previous, purely geometric methods, we further leverage a collection of training shapes to learn and constrain the reconstruction to the space of realistic shapes.

Thus, we assume that we are given a collection of shapes, each represented by a set of shape difference operators with respect to a fixed base shape. We also assume the presence of a point-wise map from the base shape to each of the shapes in the collection, which allows us to compute the "ground truth" embedding of each shape. We represent this embedding as three coordinate functions on the base shape. Our goal then is to design a network, capable of converting the input shape difference operators to the ground truth coordinate functions.

At test time, we use this network to reconstruct a target shape given only the shape difference operators with respect to the base shape. These shape difference operators can be obtained using the knowledge of a functional map from the base shape, or synthesized directly for shape analogies or interpolations applications.

Architecture To solve the problem above we developed the OperatorNet architecture, which takes as input shape difference matrices and outputs coordinate functions representing the original shapes. Our network has two modules: a shallow convolutional encoder and a 3-layer dense decoder (see Appendix, Figure B.5 for a pictorial overview).

The grid structure of shape differences is exploited by the encoder through the use of convolutions. Note however that translation invariance does not apply to these matrices.

After comparing with multiple depths of encoders, we selected a shallow version as it performed the best in practice, implying that the shape difference representation already encodes meaningful information efficiently. Moreover, as shown in [61] the edge lengths of a mesh can be recovered from intrinsic shape differences through a series of least squares problems, hinting that increasing the depth of the network and thus the non-linearity might not be necessary with shape differences.

On the other hand, the decoder is selected for its ability to transform the latent representation to coordinate functions for reconstruction and interpolation tasks.

Input Shape Differences We construct the input shape differences using a truncated eigenbasis of dimension 60 on the base shape, and the full basis on the target one, within all experiments, regardless

of the number of vertices on the actually shapes. The functional maps from the base to the targets are induced by the identity maps, since our training shapes are in 1-1 correspondence. This implies that each of the shapes is represented by three 60×60 matrices, representing the area-based, conformal and extrinsic shape differences respectively. The independence among the shape differences allows flexibility in selecting the combination of input shape differences. In Section 4.5, we compare the performance of several combinations. A detailed ablation study with respect to the input channels and network depth is presented in Appendix B.1.4. It is worth noting that recent learning-based shape matching techniques enable efficient (functional) maps estimation. In particular, we adapt the framework of [236] and evaluate OperatorNet trained with *computed* shape differences in Section 4.5.

Datasets We train OperatorNet on two types of datasets: humans and animals. For human shapes, our training set consists of 9440 shapes sampled from the DFAUST dataset [34] and 8000 from the SURREAL dataset [274], which is generated with the model proposed in [167]. The DFAUST dataset contains 4D scan of 10 human characters subject to a various of motions. On the other hand, the SURREAL dataset injects more variability to the body types. For animals, we used the parametric model proposed in SMAL [326] to generate 1800 animals of 3 different species, including lions, dogs, and horses. The meshes of the humans (resp. animals) are simplified to 1000 vertices (resp. 1769 vertices).

Loss Function OperatorNet reconstructs coordinate functions of a given training shape. Our shape reconstruction loss operates in two steps. First, we estimate the optimal rigid transformation to align the ground truth coordinate functions X_{gt} and the reconstructed ones X_{recon} using the Kabsh algorithm [22] with ground truth correspondences. Second, we estimate the mean squared error between the aligned reconstruction and the ground truth.

$$L(X_{\text{gt}}, X_{\text{recon}}) = \frac{1}{n} \sum_{i=1}^{n} (R_{X_{\text{gt}}}(X^i_{\text{recon}}) - X^i_{\text{gt}})^2. \tag{4.9}$$

Here $R_{V_{\text{gt}}}$ is the function that computes the optimal transformation between S_{recon} and S_{gt}. We align the computed reconstructions to the ground truth embedding, so that the quality of the reconstructed point cloud is invariant to rigid transformations. This is important since the shape difference operators are invariant to rigid motion of the shape, and thus the network should not be penalized, for not recovering the correct orientation. On the other hand, this loss function is differentiable, since we use a closed-form expression of $R_{X_{gt}}$, given by the SVD, which enables

back-propagation in neural network training.

4.5 Evaluation

In this section, we provide both qualitative and quantitative evaluations of the results from OperatorNet, and compare them to the geometric baselines.

Evaluation Metrics We denote by S_{gt} and S_{recon} the ground-truth and the reconstructed meshes respectively. First we denote by $d_R = L(X_{\text{gt}}, X_{\text{recon}})$, where L is the rotationally-invariant distance defined in Eq. (4.9) and X is the coordinate functions of S. For a comprehensive, unbiased evaluation and comparison, we introduce the following two new metrics: (1) $d_V = |V(S_{\text{gt}}) - V(S_{\text{recon}})|/V(S_{\text{gt}})$, i.e., the relative error of mesh volumes; (3) $d_E = \text{mean}_{(i,j)}|l_{ij}^{\text{gt}} - l_{ij}^{\text{recon}}|/l_{ij}^{gt}$, where l_{ij} is the length of edge (i, j).

Baselines Two baselines are considered: (1) the intrinsic reconstruction method from [35], in which we evaluate with the 'Shape-from-Laplacian' option and use full basis in *both* the base shape and the target shape; (2) the reconstruction method from [61], where the authors construct offset surfaces of the shapes to take account of extrinsic geometry, we evaluate with the same basis truncation as our input. Moreover, the latter also gives a version of purely intrinsic reconstruction. Beyond that, we also consider the nearest neighbourhood retrieval from the training set with respect to distances between shape difference representations.

Test Data We retain 800 shapes from DFAUST dataset as the test set, which contain 10 sub-collections (character + action sequence, each consists of 80 shapes) that are isolated from the training/validation set. For the efficiency of baseline evaluation, we further sample 5 shapes via furthest point sampling regarding the pair-wise Hausdorff distance from each of the sub-collection, resulting in a set of 50 shapes, that covers significant variability in both styles and poses in the original test set.

Quantitative Results We list all the scores regarding the metrics defined above in Table 4.1. First of all, OperatorNet using both intrinsic and extrinsic shape differences achieved the smallest reconstruction error (i.e., d_R), and the purely intrinsic version is next to the best. OperatorNet trained on shape differences from computed functional maps achieve competing performances showing that

Table 4.1: Quantitative evaluation of shape reconstruction (d_R is at the scale of 10^{-4}).

	d_R	d_V	d_E
Op.Net (Int+Ext)	**1.11**	0.014	0.045
Op.Net (Int)	2.41	0.013	0.046
Op.Net (Ext)	1.25	0.017	0.046
Op.Net (Comp)(Ext)	3.86	0.021	0.052
Op.Net (Comp)(Int+Ext)	6.22	0.022	0.053
SfL [35]	48.8	0.081	**0.012**
FuncChar [61](Int)	65.1	0.356	0.118
FuncChar [61] (Int+Ext)	28.4	0.028	0.110
NN	25.5	**0.005**	0.043

our method is efficient even in the absence of ground truth one-to-one correspondences. Note also that all versions of OperatorNet significantly outperform the other baselines.

Regarding the volume and edge recovery accuracy, either the complete or the intrinsic-only version of OperatorNet achieves second to the best result. It is worth noting that, since the nearest neighbourhood search in general retrieves the right body type, the volume is well recovered. On the other hand, full Laplacian of the target shape is provided as input for the Shape-from-Laplacian baseline, thus it is expected to preserve well the intrinsic information.

Qualitative Results We demonstrate the reconstructed shapes from OperatorNet and the aforementioned baselines in Figure 4.5, the red shape in each row therein is the respective ground-truth target shape. The base shape in this experiment (also the base shape we compute shape difference on) is shown in Figure 4.4, which is in the rest pose. The geometric baselines in general perform worse when the poses changes significantly with respect to the base (see the top two rows in Figure 4.5), but gives relatively better result when the difference is mainly on the style (see the last row). Our method, on the other hand, produces consistently good reconstruction in all the cases. It is also worth noting that, as expected, OperatorNet using all 3 types of shape differences gives both the best quantitative and qualitative results. Finally, we present a qualitative verification of the generalization power of OperatorNet in Appendix B.1.2.

4.6 Applications

In this section, we present all of our results using OperatorNet trained with both intrinsic and extrinsic shape differences, which are induced by ground-truth maps.

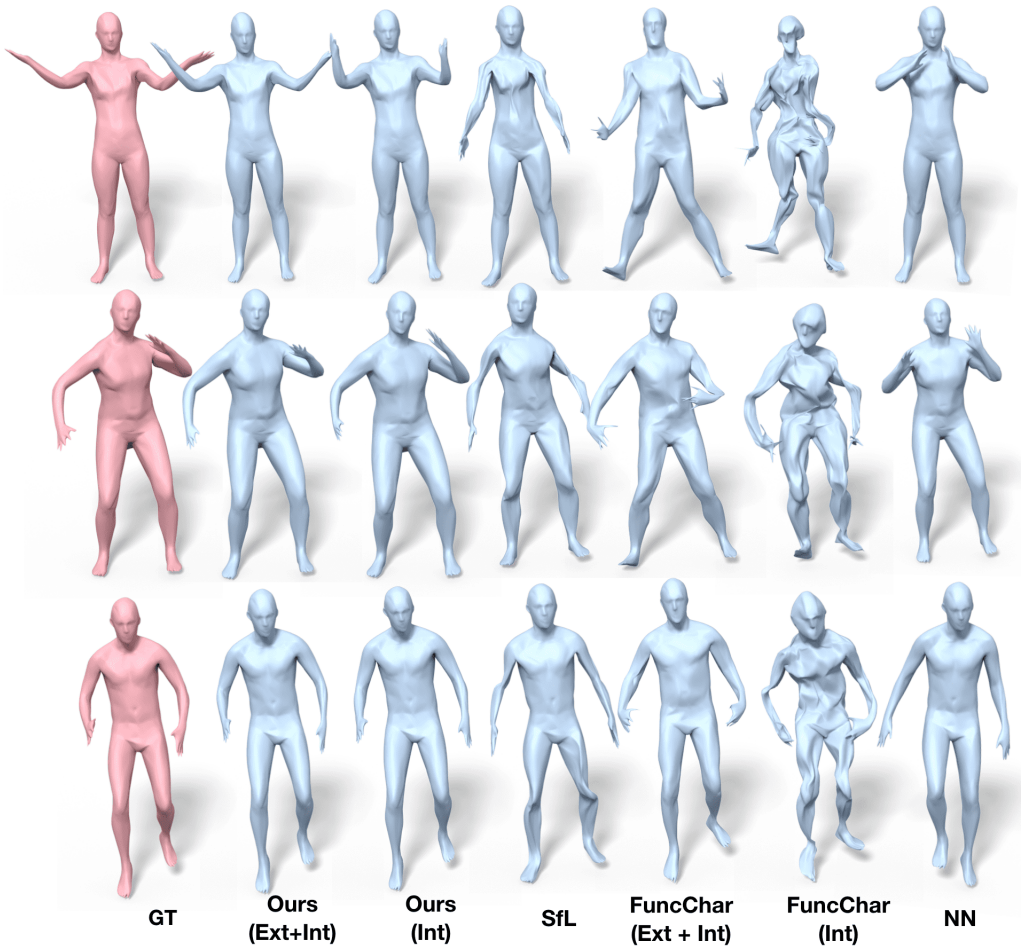


Figure 4.5: Qualitative comparison of our reconstructions and the baselines.

4.6.1 Shape interpolation

Given two shapes, we first interpolate the regarding shape differences using the formulation in Eq.(4.6) (an alternative is to interpolate shape differences linearly, we provide comparison to such in Appendix B.1.3), and then synthesize intermediate shapes by inferring the interpolated shape differences with OperatorNet.

We compare our method against several baselines. First, a PointNet autoencoder is trained with the encoder architecture from [219] and with our decoder. Two versions of PointNet are trained: one autoencoder with spatial transformers and one without. The autoencoder without spatial transformers performs better at reconstruction and interpolation, it is therefore selected for the comparisons. Another autoencoder based on PointNet++ [222] is similarly trained.

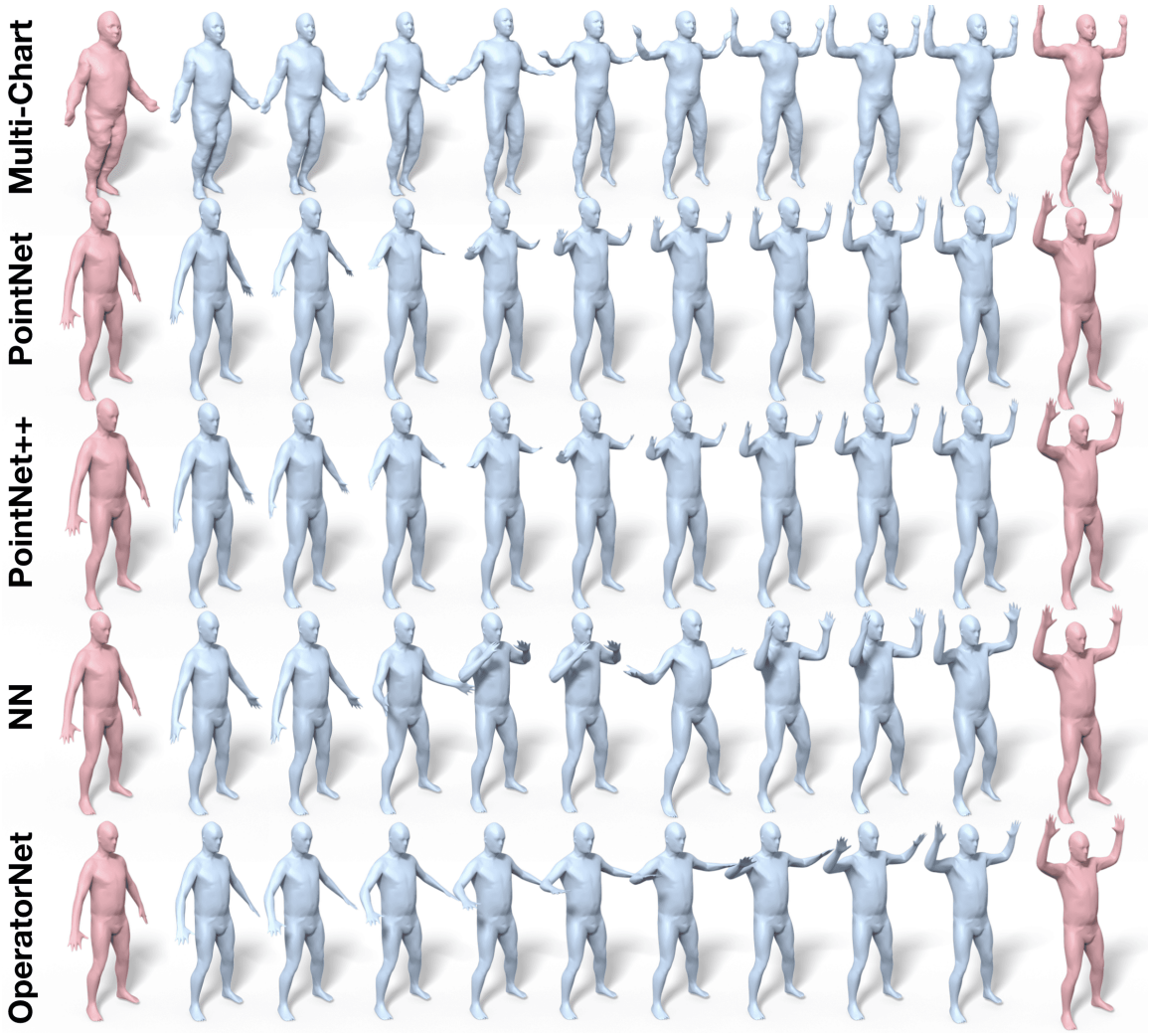


Figure 4.6: Shape interpolation between two humans. Note that the interpolation by Multi-chart GAN is less evenly developed than the bottom row (e.g., the positions of the arms in the centre three shapes change abruptly); autoencoders based on PointNet and PointNet++ both produce shapes with local area distortion; and the interpolation from Nearest-Neighborhood retrieval is not continuous. Contrastingly, the interpolation via OperatorNet is more natural and smooth, compared to the baselines (see text for more details).

Nearest neighbor (NN) interpolation retrieves the nearest neighbour of the interpolated shape differences in the training set and uses the corresponding labels for interpolation.

Moreover, we also compare to the interpolation result from a recent work [29], where a GAN is trained to generate realistic human shapes. Note that, as stated in [29], the interpolation is done as follows: first one picks two randomly generated latent vectors z_1, z_2, which, via the GAN gives

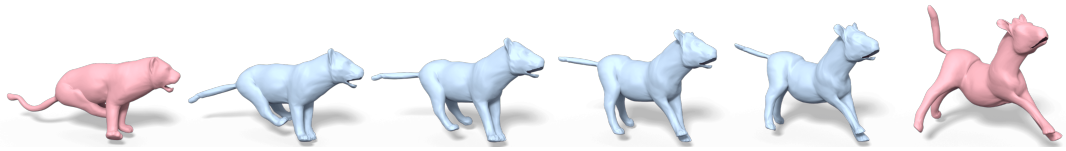


Figure 4.7: Shape interpolation from a tiger (left) to a horse (right) using OperatorNet trained on animals dataset.

arise to two shapes $G(z_1), G(z_2)$. Then, the interpolation between the two shapes are achieved as $G(z(t))$, where $z(t) = (1-t)z_1 + tz_2$. In particular, we randomly generate 1000 shapes using their trained model and pick $G(z_i), i = 1, 2$ that are nearest to the red shapes in the last row of Figure 4.6 for interpolation comparison.

In Figure 4.6, the first row shows the interpolation between the two end shapes by [28], which is not developed evenly, for instance, the arms change abruptly during the three middle shapes, while there is little change on that region afterwards. As shown in the second and the third rows, both the results of the two autoencoders suffer obvious area distortions in the arms (see a detailed comparison in Figure 4.1). The fourth row of Figure 4.6 indicates the NN approach fails to deliver a continuous deformation sequence. In contrast, interpolation using OperatorNet is continuous and respects the structure and constraints of the body, suggesting that shape differences encode efficiently the structure.

We also train OperatorNet on the animals dataset as described in Section 4.4 and show in Figure 4.7 an interpolation from a tiger to a horse.

4.6.2 Shape analogy

Our second application is to construct semantically meaningful new shapes based on shape analogies. Given shapes S_A, S_B, S_C, our goal is to construct a new shape S_X, such that S_C is to S_X as S_A is to S_X.

Following the discussion in Section 2.4, the functoriality of shape differences allows an explicit and mathematically meaningful way of constructing the shape difference of S_X, given that of S_A, S_B and S_C. Namely, $\mathbf{D}_X = \mathbf{D}_C \mathbf{D}_A^{+} \mathbf{D}_B$. Then, with our OperatorNet, we reconstruct the embedding of the unknown S_X by feeding $\mathbf{D}_X$ to the network.

We compare our results to that of the PointNet autoencoder. For the latter, we reconstruct S_X by decoding the neural-network's latent code with respect to the additive formula, i.e., $l_X = l_C - l_A + l_B$, where l_A is the latent code of shape S_A (and similarly for S_B, S_C).

In Figure 4.9, we show shape analogies of pose transfer (top row) and style transfer (bottom row) via OperatorNet and PointNet autoencoder on human shapes. It is evident that our results are both more natural and intuitive.

We also show analogies among animals in Figure 4.10, where we present both pose transfer (top row) and style transfer (bottom row) and comparison to the results of PointNet.

Moreover, we present a set of more challenging shape analogies that transfer gender across human shapes in Figure 4.8. Namely, we fix S_A, S_B, and test analogies with different S_C (the 6 male shapes in red). Note that S_A and S_B are two characters in comparable poses and styles but of different genders, ideally the right analogies, S_X, should be a 'female' version of S_C with similar poses and styles.

We present the analogies from OperatorNet in the first results column, and those from PointNet in the second results column (both in blue). It is obvious that PointNet, though works in some cases, in general produces less semantically meaningful analogies than ours (see discrepancies in the red dotted boxes).

4.7 Conclusion

In this chapter we have introduced a novel learning-based technique for recovering shapes from their difference operators. Our key observation is that shape differences, stored as compact matrices lend themselves naturally to learning and allow to both recover the underlying shape space in a collection and encode the geometry of individual shapes. We also introduce a novel extrinsic shape difference operator and show its utility for shape reconstruction and other applications such as shape interpolation and analogies. The approach is only well-adapted to shapes represented as triangle meshes. A valuable future work would be to extend this framework to both learn the optimal inner products from data, and adapt the pipeline to other shape representations, such as point clouds or triangle soups.

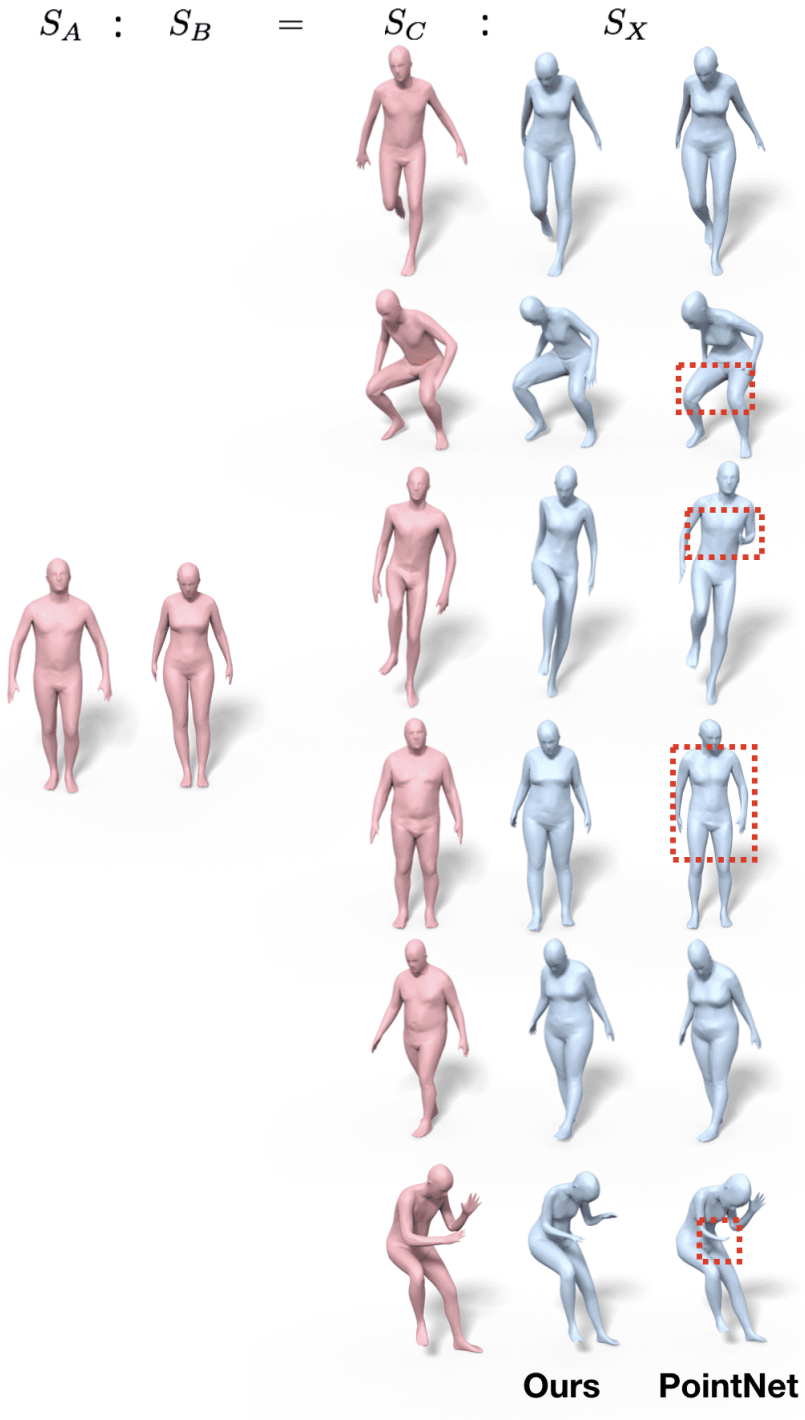


Figure 4.8: Transferring gender via shape analogies: S_A and S_B are a fixed pair of human shapes with similar poses and styles, but of different genders. We generate S_X, which is supposed to be a 'female' version of the varying S_C. Our analogies are semantically meaningful, while PointNet can produce suboptimal results (see the red dotted boxes for the discrepancies).

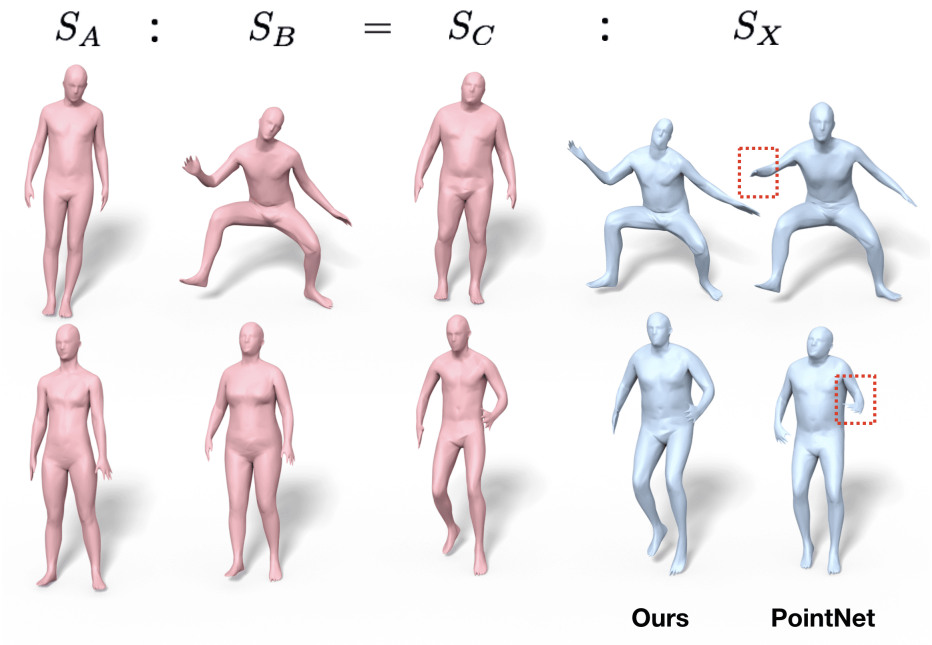


Figure 4.9: Human shape analogies via OperatorNet and PointNet autoencoder (see the red dotted boxes for the discrepancies).

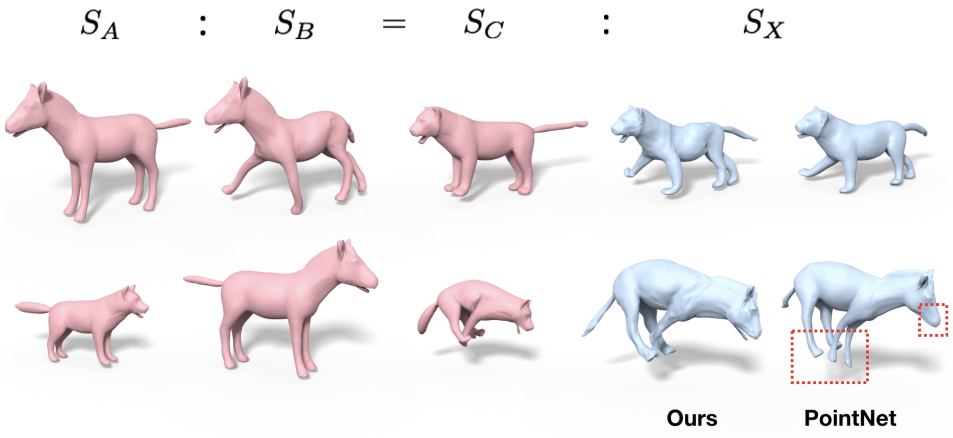


Figure 4.10: Animal shape analogies via OperatorNet and PointNet autoencoder (see the red dotted boxes for the discrepancies).

Chapter 5

Discriminating the Shape of Objects with Referential Language

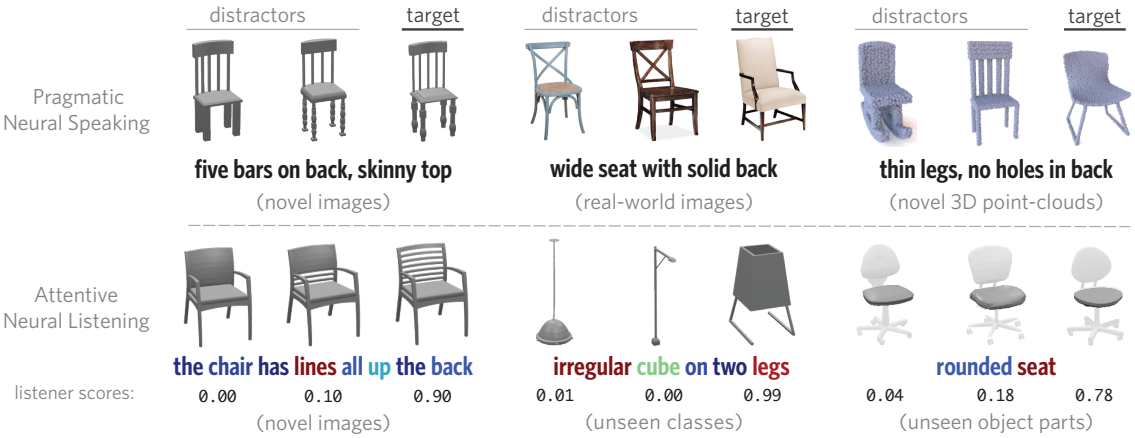


Figure 5.1: **Teaser summary of Chapter's contributions.** We introduce a novel corpus of utterances that refer to the shape of objects and use it to develop multimodal neural speakers and listeners with broad generalization capacity. **Top row**: Our neural speaker generates utterances to distinguish a 'target' shape from two 'distractor' shapes in unseen: images of synthetic data (left), *out-of-distribution* (OOD) real-world images (center), and 3D point-clouds of CAD models (right). **Bottom row**: Our neural listener interprets human-generated utterances in unseen (left-to-right): images of synthetic data, OOD object *classes* (here, lamps), and OOD isolated object *parts*. Listener scores indicate the model's confidence about which object the utterance refers to. The words are color-coded according to their importance, as judged by the attention module of this listener (warmer color indicates higher attention).

5.1 Overview

In this chapter we explore how **fine-grained differences** between the shapes of common objects are expressed in **language**, grounded on 2D and/or 3D object representations. We first introduce a large scale, carefully controlled dataset of human utterances each of which refers to a 2D rendering of a 3D CAD model so as to distinguish it from a set of shape-wise similar alternatives. Using this dataset, we then develop neural language understanding (listening) and production (speaking) models that vary in their grounding (pure 3D forms via point-clouds vs. rendered 2D images), the degree of pragmatic reasoning captured (e.g. speakers that reason about a listener or not), and the neural architecture (e.g. with or without attention). We show that these models perform well with both synthetic and human partners, and with held out utterances and objects. We also show that these models are capable of **zero-shot** transfer learning to novel object classes (e.g. transfer from training on chairs to testing on lamps), as well as to real-world images drawn from furniture catalogs. Finally, with carefully done lesion studies we demonstrate that the neural listeners depend heavily on **part-related words** and associate these words correctly with **visual parts of objects**. Interestingly, this part-awareness is acquired without any explicit supervision on semantic shape parts (like those used in Chapter 3) and furthermore transfer learning to novel classes is most successful when known part-related words are available. Finally, unlike the material presented in Chapter 4, here we are exploring shape-differences from a more **human-centric** angle by focusing on the cognitive aspects of shape differences as these are expressed in language. The materials of this chapter are based on our paper *"ShapeGlot: Learning Language for Shape Differentiation"* [7] and taken together they illustrate a practical approach to language grounding, and a case study in the relationship between object shape and linguistic structure when it comes to object differentiation.

5.2 Introduction

Objects are best understood in terms of their structure and function, both of which rest on a foundation composed of object parts and their relations [79, 77, 324, 75]. Natural language has been optimized across human history to solve the problem of efficiently communicating the aspects of the world most relevant to one's current goals [133, 82]. As such, language can provide an effective medium to describe the *shape* and the *parts* of different objects, and as a result, to express *object differences*. For instance, when we see a chair we can analyze it into semantically meaningful parts, like its *back* and its *seat*, and can combine words to create utterances that reflect its geometric and topological

shape-properties e.g. 'has a wide seat with a solid back'. Moreover, given a specific communication context, we can craft references that are not merely true, but which are also relevant e.g. we can refer to the lines found in a chair's back to *distinguish* it among other similar objects (see Fig. 5.1).

In this chapter we explore this interplay between natural, referential language, and the shape of common objects. While a great deal of recent work has explored visually-grounded language understanding [126, 194, 312, 170, 169, 311], the resulting models have limited capacity to reflect the geometry and topology (i.e. the shape) of the underlying objects. This is because reference in previous studies was possible using properties like the object's *color*, or spatial configuration, including the absolute or relative (to other objects) *location*. Indeed, eliciting language that refers only to shape properties requires carefully controlling the objects, their presentation, and the underlying linguistic task. To address these challenges, in this work we utilize 3D CAD representations of objects which allow for flexible and *controlled* presentation (i.e. textureless, uniform-color objects, viewed in a fixed pose). We further make use of the 3D form to construct a reference game task in which the referred object is *shape-wise* similar to the distracting objects. The result of this effort is a new multimodal dataset, termed *ShapeGlot*, comprised of 4,511 unique chairs from ShapeNet [46] and 78,789 referential utterances. In ShapeGlot chairs are organized into 4,054 sets of size 3 (representing communication contexts) and each utterance is intended to distinguish a chair in context.

We use ShapeGlot to build and analyze a pool of modern neural language understanding (listening) and production (speaking) models. These models vary in their grounding (pure 3D forms via point-clouds vs. rendered 2D images), the degree of pragmatic reasoning captured (e.g. speakers that reason about a listener or not) and their precise neural architecture (e.g. with or without word attention, with context-*free*, or context-*aware* object encodings). We evaluate the effect of these choices on the original reference game task with both synthetic and human partners and find models with strong performance. Since language conveys abstractions such as object parts, that are shared between object categories, we hypothesize that our models learn robust representations that are *transferable* to objects of unseen classes (e.g. training on chairs while testing on lamps). Indeed, we show that these models have strong generalization capacity to novel object *classes*, as well as to *real-world* images drawn from furniture catalogs.

Finally, we explore *how* our models are succeeding on their communication tasks. We demonstrate that the neural listeners learn to prioritize the same abstractions in objects (i.e. properties of chair parts) that humans do in solving the communication task, despite *never* being provided with an explicit decomposition of these objects into parts. Similarly, we find that neural listeners transfer to novel object classes more successfully when known part-related words are available. Finally,

we show that *pragmatic* neural speakers who consult an imagined (simulated) listener produce significantly more informative utterances than listener-unaware, *literal* speakers, as measured by human performance in identifying the correct object given the generated utterance.

5.3 Related Work

Image labeling and captioning Our work builds on recent progress in the development of vision models that involve some amount of language data, including object categorization [251, 319] and image captioning [122, 278, 301]. Unlike object categorization, which pre-specifies a fixed set of class labels to which all images must project, our systems use open-ended, referential language. Similarly to other recent works in image captioning [177, 194, 312, 170, 276, 169, 311], instead of captioning a single image (or entity therein), in isolation, our systems learn how to communicate across diverse communication contexts.

Reference games In our work we use reference games [126] in order to operationalize the demand to be relevant in context. The basic arrangement of such games can be traced back to the language games explored by Wittgenstein [289] and Lewis [153]. For decades, such games have been a valuable tool in cognitive science to quantitatively measure inferences about language use and the behavioral consequences of those inferences [233, 137, 57, 272]. Unlike traditional captioning or labeling tasks where participants produce labels in isolation, reference games have a well-defined task objective and can be easily designed to elicit a wide variety of referring expressions even for the same target object by manipulating which objects serve as the distractors. We exploit these properties to ensure that our system acquires flexibility in how it combines linguistic and visual representations to solve the task. Recently, these approaches have also been adopted as a benchmark for discriminative or context-aware NLP [203, 15, 259, 276, 193, 58, 148].

Rational speech acts framework Our models draw on recent formalization of human language use in the Rational Speech Acts (RSA) framework [88]. At the core of RSA is the Gricean proposal [93] that speakers are agents who select utterances that are parsimonious yet informative about the state of the world. RSA formalizes this notion of informativity as the expected reduction in the uncertainty of an (internally simulated) listener, as our pragmatic speaker does. The literal listener in RSA uses semantics that measure compatibility between an utterance and a situation, as our baseline listener does. Previous work has shown that RSA models account for context sensitivity in speakers and listeners [92, 193, 313, 80]. Our results add evidence for the effectiveness of this approach in the shape domain.

5.4 The ShepeGlot Dataset & Task

ShapeGlot consists of triplets of chairs coupled with referential utterances that aim to distinguish one chair (the 'target') from the remaining two (the 'distractors'). To obtain such utterances, we paired participants from Amazon's Mechanical Turk (AMT) to play an online reference game [98]. On each round of the game the two players were shown the same triplet of chairs. The designated target chair was privately highlighted for one player (the 'speaker') who was asked to send a message through a chat box such that their partner (the 'listener') could successfully select it. To ensure speakers used *only* shape-related information, we scrambled the positions of the chairs for each participant independently and used textureless, uniform-color renderings of pre-aligned 3D CAD models, taken from the same viewpoint. To ensure that the communicative interaction was natural, no constraints were placed on the chat box: referring expressions from the speaker were occasionally followed by clarification questions from the listener or other discourse.

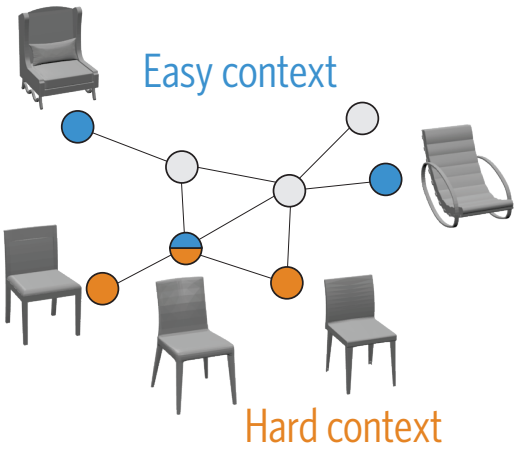


Figure 5.2: Constructing "hard" and "easy" contexts by exploiting the latent *shape* similarity of 3D chair point clouds. The bi-colored node with cyan/orange marks a canonical chair (with high in-degree in the neighbor-graph). Cyan or orange-colored are its selected distractors in each context type.

A key decision in building our dataset concerned the construction of contexts that would reliably elicit *diverse* and potentially *very* fine-grained contrastive language. To achieve diversity we considered all ∼7,000 chairs from ShapeNet. This object class is geometrically complex, highly diverse, and abundant in the real world. To control the granularity of fine-grained distinctions that were necessary in solving the communication task, we constructed two types of contexts: *hard* contexts consisted of very similar shape-wise chairs, and *easy* contexts consisted of less similar chairs. To measure shape-similarity in a scalable manner, we used the semantically rich latent space of the Point Cloud-AutoEncoder (PC-AE) we presented in Chapter 2. We note, that point-clouds are an intrinsic representation of a 3D object, *oblique* to color or texture. After extracting a 3D point-cloud from the surface of each ShapeNet model we computed the underlying K-nearest-neighbor graph among all models according to their PC-AE embedding distances. For a chair with sufficiently high-in degree on this graph (corresponding intuitively to a canonical chair) we contrasted it with four distractors:

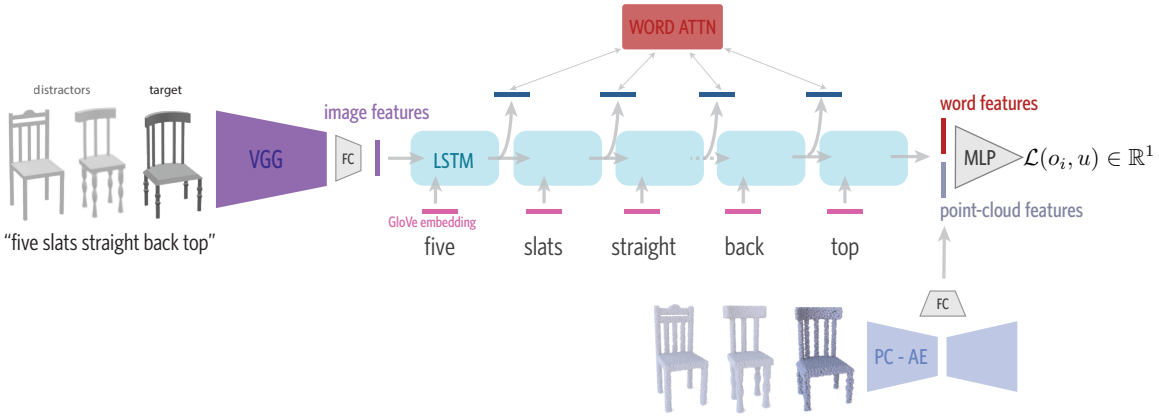


Figure 5.3: *Baseline* listener architecture combining 2D images, 3D point-clouds and linguistic utterances.

the two *closest* to it in latent-space, and two that were sufficiently far (see Appendix C for details). Last, we note that we *counterbalanced* the collected utterances, by considering every chair in a given context as the context's target (in different games).

Before we present our neural agents, we identify some distinctive properties of our corpus. Human performance on the reference game was high, but listeners made significantly more errors in the hard contexts (accuracy 94.2% vs. $97.2\%, z = 13.54, p < 0.001$). Also, in hard contexts longer utterances were used to describe the targets (on average 8.4 words vs. 6.1, $t = -35, p < 0.001$). A wide spectrum of descriptions was elicited, ranging from more holistic/categorical (e.g. 'the rocking chair') common for easy contexts, to more complex and fine-grained language, (e.g. 'thinner legs but without armrests') common for hard ones. Interestingly, 78% of the produced utterances contained at least one part-related word: *back*, *legs*, *seat*, *arms*, or closely related synonyms e.g. *armrests*.

5.5 Building Neural Listeners for Shape-based Reasoning

Developing neural listeners that reason about shape-related properties is a key contribution of our work. Below we conduct a detailed comparison between three distinct architectures, highlight the effect of different regularization techniques, and investigate the merits of different representations of 3D objects for the listening task, namely, 2D rendered images and 3D surface point clouds. In what follows, we denote the three objects of a communication context as $O = \{o_1, o_2, o_3\}$, the corresponding word-tokenized utterance as $U = u_1, u_2, \ldots$ and as $t \in O$ the designated target.

Our proposed listener is inspired by [193]. It takes as input a (latent code) vector that captures

shape information for each of the objects in O, and a (latent code) vector for each token of U, and outputs an *object–utterance* compatibility score $\mathcal{L}(o_i, U) \in [0, 1]$ for each input object. At its core lies a multi-modal LSTM [106] that receives as initial input ("is grounded" with) the vector corresponding to one object, processes the word-sequence U, and is read out by an MLP to yield a single number (a compatibility score). This is *repeated* for each object, while *sharing* all network parameters across the objects. The resulting three scores are soft-max normalized and compared to the ground-truth indicator vector of the target, under the cross-entropy loss.*

Shape encoders We experiment with three representations to capture the shapes of the underlying objects: (a) the bottleneck vector of a pretrained Point Cloud-AutoEncoder (PC-AE), (b) the embedding provided by a convolutional network operating on single-view images of non-textured 3D objects, or (c) a combination of (a) and (b). Specifically, for (a) we use the PC-AE architecture we introduced in Chapter 2 trained with single-class point clouds extracted from the surfaces of 3D CAD models, while for (b) we use the activations of the penultimate layer of a VGG-16 [251], pre-trained on ImageNet [68], and fine-tuned on an 8-way classification task with images of objects from ShapeNet. For each representation we project the corresponding latent code vector to the input space of the LSTM using a fully connected (FC) layer with L_2-norm weight regularization. While there are many ways to incorporate image-based with point-cloud based features in the LSTM, we found that the best performance occurs when we i) ground the LSTM with the image-based codes, ii) concatenate the LSTM's output (after processing U) with the point cloud-based codes, and iii) feed the concatenated result in a shallow MLP that produces the compatibility score (see Supp. for a *visual overview* of the pipeline and more details). We note that proper regularization is *critical*: adding dropout at the input layer of the LSTM and L_2 weight regularization and dropout at and before the FC projecting layers improves performance $\sim 10\%$.

Incorporating context information Our baseline listener architecture (*Baseline*, just described) first scores each object *separately* then applies softmax normalization to yield a score distribution over the three objects. We also consider two alternative architectures that explicitly encode information about the *entire* context before scoring a single object. The first alternative (*Early-Context*), is identical to the proposed architecture, except for the codes used to *ground* the LSTM. Specifically, if v_i is the image-based code vector of the i-th object, instead of using v_i as the grounding vector for o_i, a shallow convolutional network is introduced to create a more complex (context-aware) feature. This network, of which the output *is* the grounding code for o_i, receives the signal $f(v_j, v_k)||g(v_j, v_k)||v_i$, where f, g are the symmetric max/mean-pool functions,

*Architecture details and hyper-parameters for all the experiments, are provided in the Appendix C.

|| denotes feature-wise concatenation and v_j, v_k are the codes of the remaining objects. Here, we use symmetric functions to induce the orderless nature of our contexts. The second alternative (*Combined-Interpretation*) inputs the image-based code vectors for *all* three objects sequentially to the LSTM and then proceeds to process the tokens of U *once*, before yielding three scores. Similarly to the *Baseline* architecture, point clouds are incorporated in both alternatives at the MLP operating after the LSTM.

Word attention We hypothesized that a listener forced to prioritize a few tokens in each utterance would learn to prioritize tokens that express properties that distinguish the target from the distractors (and, thus, perform better). To test this hypothesis, we augment the listener models with a standard *bilinear attention mechanism* [247]. Specifically, to estimate the 'importance' of each token u_i we compare the output of the LSTM when it inputs u_i (denoting the output as r_i); with the hidden state *after* the entire utterance has been processed (denoted as h). The relative importance of each token is $a_i \triangleq r_i^T \times W_{\text{att}} \times h$, where W_{att} is a trainable diagonal matrix. The new (weighted) output of the LSTM is: $\sum_{i=1}^{|U|} r_i \odot \hat{a}_i$, where $\hat{a}_i = \frac{\exp(a_i)}{\sum_j^{|U|} \exp(a_j)}$ and $\odot$ is the point-wise product.

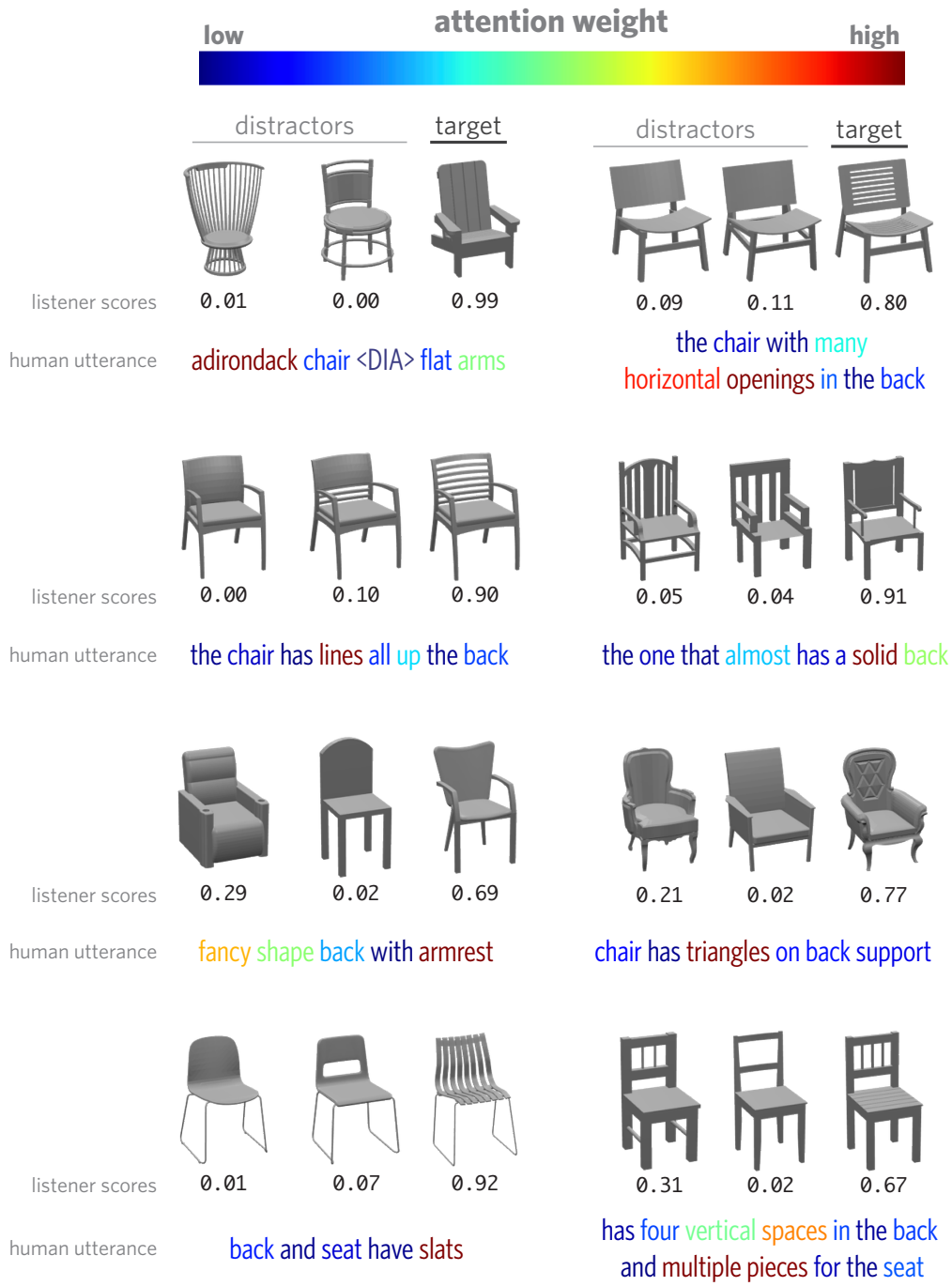
5.6 Listening Comprehension Experiments

Architecture	Overall	Subpopulations		
		Hard	Easy	Sup-Comp
Combined-Interpretation	$75.9 \pm 0.5\%$	$67.4 \pm 1.0\%$	$83.8 \pm 0.6\%$	$74.4 \pm 1.5\%$
Early-Context	$79.4 \pm 0.8\%$	**70.1** $\pm 1.3\%$	$88.1 \pm 0.6\%$	$75.6 \pm 2.2\%$
Baseline	**79.6** $\pm 0.8\%$	$69.9 \pm 1.3\%$	**88.8** $\pm 0.4\%$	**76.3** $\pm 1.3\%$

Table 5.1: Comparing different ways to include context. The simplest *Baseline* model performs as well as more complex alternatives. Subpopulations are the subsets of test data containing: hard contexts (shape-wise similar distractors), easy contexts, superlatives or comparatives.

We begin our evaluation of the proposed listeners using two reference tasks based on different data splits. In the *language generalization* task, we test on target objects that were seen as targets in at least one context during training but ensure that all utterances in the test split are from unseen speakers. In the more challenging *object generalization* task, we restrict the set of objects that appeared as targets in the test set to be *disjoint* from those in training such that all speakers *and* objects in the test split are new. For each of these tasks, we evaluate choices of input modality and word attention, using $[80\%, 10\%, 10\%]$ of the data, for training, validating and testing purposes.

Figure 5.4: **Examples of attention weights on human utterances**. The listener's LSTM appears to learn attention weights that emphasize the more informative words disambiguating the referent. For these results the *Baseline* listener is used and the attention-scores are extracted when the target object is grounding the LSTM.



	Input Modality	Language Task	Object Task
No Attention	Point Cloud	$67.6 \pm 0.3\%$	$66.4 \pm 0.7\%$
	Image	$81.2 \pm 0.5\%$	$77.4 \pm 0.7\%$
	Both	$83.1 \pm 0.4\%$	$78.9 \pm 1.0\%$
With Attention	Point Cloud	$67.4 \pm 0.3\%$	$65.6 \pm 1.4\%$
	Image	$81.7 \pm 0.5\%$	$77.6 \pm 0.8\%$
	Both	**83.7** $\pm 0.3\%$	**79.6** $\pm 0.8\%$

Table 5.2: Performance of the *Baseline* listener architecture using different object representations and with/without word level attention, in two reference tasks.

Baseline listener accuracies are shown in Table 5.2.[†] Overall the *Baseline* achieves good performance. As expected, the listeners have higher accuracy on the language generalization task (3.2% on average). The attention mechanism on words yields a mild performance boost, as long as images are part of the input. Interestingly, images provide a significantly better input than point-clouds when only one modality is used. This may be due to the higher-frequency content of images (we use point-clouds with only 2048 points), or the fact that VGG was pre-trained while the PC-AE was not. However, we find *significant* gains in accuracy (4.1% on average) from exploiting the two object representations *simultaneously*, implying a complementarity among them.

Next, we evaluate how the different approaches in incorporating context information described in Section 5.5 affect listener performance. We focus on the more challenging object generalization task, using listeners that include attention and both object modalities. We report the findings in Table 5.1. We find that the *Baseline* and *Early-Context* models perform best overall, outperforming the *Combined-Interpretation* model, which does not share weights across objects. This pattern held for both hard and easy contexts of our dataset. We further explore the small portion ($\sim 14\%$) of our test set that use explicitly contrastive language: superlatives ('skinniest') and comparatives ('skinnier'). Somewhat surprisingly we find that the *Baseline* architecture remains competitive against the architectures with more explicit context information. The *Baseline* model thus achieves high performance and is the most flexible (at test time it can be applied to *arbitrary-sized* contexts); we focus on this architecture in the explorations below.

[†]In all results mean accuracies and standard errors across 5 random seeds are reported, to control for the data-split populations and the initialization of the neural-network.

5.6.1 Exploring learned representations

Linguistic ablations Which aspects of a sentence are most critical for our listener's performance? To inspect the properties of words receiving the most attention, we ran a part-of-speech tagger on our corpus. We found that the highest attention weight is placed on *nouns*, controlling for the length of the utterance. However, adjectives that *modify* nouns received more attention in hard contexts (controlling for the average occurrence in each context), where nouns are often not sufficient to disambiguate (see Fig. 5.5A). To more systematically evaluate the role of higher-attention tokens in listener performance, we conducted an utterance lesioning experiment. For each utterance in our dataset, we successively replaced words with the <UNK> token according to three schemes: (1) from highest attention to lowest, (2) from lowest attention to highest, and (3) in random order. We then fed these through an equivalent listener trained *without* attention. We found that up to 50% of words can be removed without much performance degradation, but only if these are low attention words (see Fig. 5.5B). Our word-attentive listener thus appears to rely on context-appropriate content words to successfully disambiguate the referent.

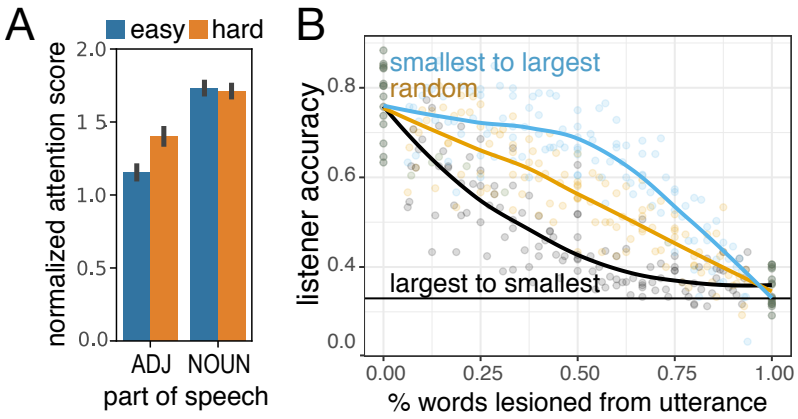


Figure 5.5: (A) The listener places more attention on adjectives in hard (orange) triplets than easy (blue) ones. The histogram's heights depict mean attention scores normalized by the length of the underlying utterances; the error bars are bootstapped 95% confidence intervals. (B) Lesioning highest attention words to lowest worsens performance more than lesioning random words or lesioning lowest attention words.

Visual ablations To test the extent to which our listener is relying on the same semantic *parts* of the object as humans, we next conducted a lesion experiment on the visual input. We took the subset of our test set where (1) all chairs had complete part annotations available [308] and (2) the

corresponding utterance mentioned a *single* part (17% of our test set). We then created lesioned versions of all three objects on each trial by removing pixels of images (and/or points when point-clouds are used), corresponding to parts according to two schemes: *removing* a single part or *keeping* a single part. We did this either for the mentioned one, or another part, chosen at random. We report listener accuracies on these lesioned objects in Table 5.3. We found that removing random parts hurts the accuracy by 10.4% on average, but removing the mentioned part dropped accuracy more than three times as much, nearly to chance. Conversely, keeping *only* the mentioned part while lesioning the rest of the image merely drops accuracy by 10.6% while keeping a non-mentioned (random) part alone brings accuracy down close to chance. In other words, on trials when participants depended on information about a part to communicate the object to their partner, we found that visual information about that part was both *necessary and sufficient* for the performance of our listener model.

	Single Part Lesioned	**Single Part Present**
Mentioned Part	$42.8\% \pm 2.3$	$66.8\% \pm 1.4$
Random Part	$67.0\% \pm 2.9$	$38.8\% \pm 2.0$

Table 5.3: Evaluating the part-awareness of neural listeners by lesioning object *parts*. Results shown are for image-only listeners, with average accuracy of 77.4% when *intact* objects are used. Similar findings regarding point-cloud-based listeners are provided in the Appendix C.

5.7 Building Pragmatic Neural Speakers for Shape Reference

5.8 Neural Speakers

Architecture Next, we explore models that learn to generate an utterance that refers to the target and which distinguishes it from the distractors. Similarly to a neural listener the heart of these (speaker) models is an LSTM which encodes the objects of a communication context, and then decodes an utterance. Specifically, for an *image-based* speaker, on the first three time steps, the LSTM input is the VGG code of each object. Correspondingly, for a *point-cloud-based* speaker, the LSTM's initial input is the object codes extracted from a PC-AE. During training and after the object codes are processes by the LSTM, the LSTM receives sequentially the i-th utterance token, while at its output if forced to predict the $(i+1)$-th token (i.e. we use teacher-force [288]). For these models we feed the target object always last (third), eliminating the need to represent an index indicating the target's position. To find the best model hyper-parameters (e.g. L_2-weights, dropout-rate and # of

LSTM neurons) and the optimal amount of training, we sample synthetic utterances from the model during training and use a pretrained *listener* to select the result with the highest listener accuracy. We found this approach to produce results that yield better quality utterances than evaluating with listening-unaware metrics like BLEU [204].

Variations The above (*literal*) speakers can learn to generate language that discriminates targets from distractors. To test the degree to which distractor objects are used for this purpose, we experimented with *context-unaware* speakers that were provided with the latent code of the target *only*, (and are otherwise identical to the above *literal* models). Furthermore, and motivated by the recursive social reasoning characteristic of human pragmatic language use (as formalized in the Rational Speech Act framework [88]), we created *pragmatic* speakers that choose utterances according to their capacity to be discriminative, as judged by a pretrained 'internal' listener. In this case, we sample utterances from the (*literal*) speakers, but score (i.e. re-rank) them with:

$$\beta \log(P_L(t|U,O)) + \frac{(1-\beta)}{|U|^\alpha} \log(P_S(U|O,t)), \tag{5.1}$$

where P_L is the listener's probability to predict the target (t) and P_S is the likelihood of the *literal* speaker to generate U. The parameter α controls a length-penalty term to discourage short sentences [291], while β controls the relative importance of the speaker's vs. the listener's opinions.

5.9 Speaker Experiments

Qualitatively, our speakers produce good object descriptions, see Fig. 5.6 for examples, with the pragmatic speakers yielding more discriminating utterances.[‡] To quantitatively evaluate the speakers we measured their success in reference games with two different kinds of partners: with independently-trained neural listeners and with human listeners. To conduct a *fair* study when we used a neural listener for evaluations, we split the training data in half. The evaluating listener was trained using one half, while the 'internal' listener used by the pragmatic speaker was trained on the remaining half. For the human-based evaluations, we first used the *literal* and *pragmatic* variants to generate an utterance for every context of the test split of the *object-generalization* task (which contains 1200 unique contexts). We then showed the resulting utterances to participants recruited with AMT and asked them to select the object from context that the speaker was referring to. We collected

[‡]The project's webpage contains additional qualitative results.

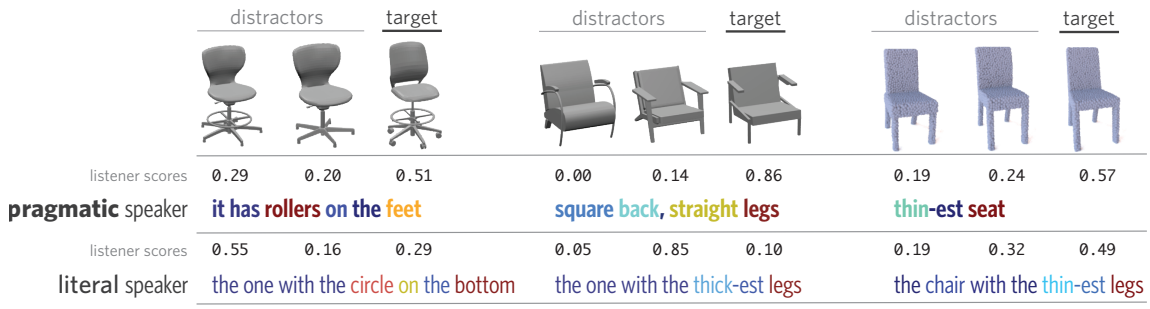


Figure 5.6: *Pragmatic* vs. *literal* speakers in unseen ('hard') contexts. The pragmatic generations successfully discern the target even in cases where the literal generations fail. The left and center contexts (gray-color) are used by image-based speakers/listeners, and the right-most by point-cloud-based ones. The utterances are color-coded according to the attention placed by a separate evaluating neural listener whose classification scores are shown above each corresponding utterance.

approximately 2.2 responses for each context. Here, we used the synthetic utterances with the highest scores (Eq. 5.1) from each model, with optimal (per-validation) α and an 'aggressive' $\beta = 1.0$. We note that while the *point-based* speakers operate *solely* with 3D point-clouds, we sent their generated utterances to AMT coupled with CAD rendered images, so as to keep the visual (AMT-human) presentation identical across the two variants.

Table 5.4: Evaluating neural speakers operating with 3D point-cloud or image-based object representations, across architectural variants.

Speaker Architecture	**Modality**	**Neural Listener**	**Human Listener**
Context Unaware	Point Cloud	$59.1 \pm 2.0\%$	-
	Image	$64.0 \pm 1.7\%$	-
Literal	Point Cloud	$71.5 \pm 1.3\%$	66.2
	Image	$76.6 \pm 1.0\%$	68.3
Pragmatic	Point Cloud	$90.3 \pm 1.3\%$	69.4
	Image	**92.2** $\pm 0.5\%$	**78.7**

We found (see Table 5.4) that our *pragmatic* speakers perform best with both neural and human partners. While their success with the neural listener model may be unsurprising, given the architectural similarity of the internal listener and the evaluating listener, *human* listeners were 10.4 percentage points better at picking out the target on utterances produced by the *pragmatic* vs. *literal* speaker for the best-performing (*image-based*) variant. Similar to what we saw in the listener experiments (Section 5.6), we found that (sole) point-cloud-based speakers achieve lower performance than image-based variants. However, we also found an asymmetry between the listening

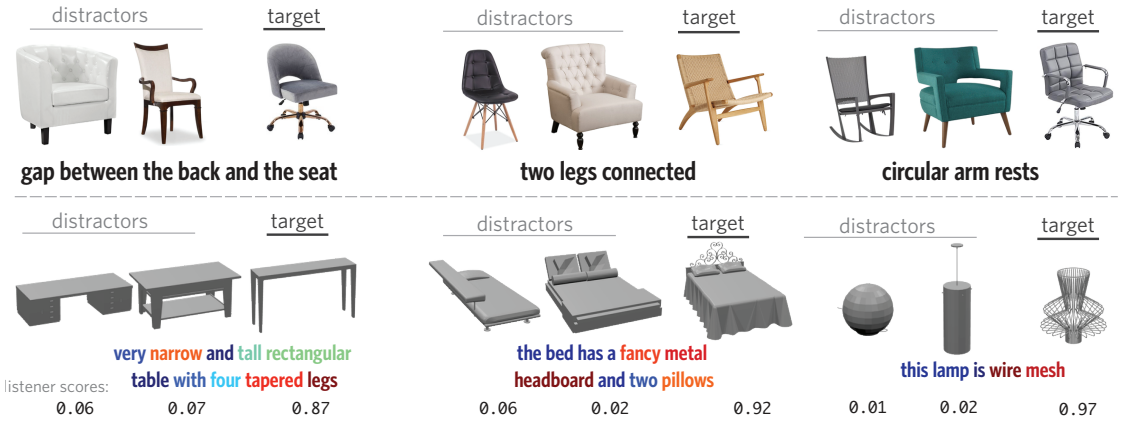


Figure 5.7: Examples of *out-of-distribution* neural speaking and listening. **Top row**: model generations for *real-world* catalogue images. The speaker successfully describes fine grained shape differences on images with rich color and texture content; two factors *not* present in the training data. **Bottom row**: results of applying a word-attentive listener on renderings of CAD objects from *unseen* classes with human-produced utterances. The listener can detect the (often localized) visual cues that humans refer to, despite the large visual discrepancy of these objects from the training-domain of chairs. (The utterances are color coded according to the attention placed to them by the attentive neural listener.)

and speaking tasks: while context-unaware (*Baseline*) listeners achieved high performance, we found that context-unaware speakers fare significantly worse than context-aware ones. Last, we note that both literal and pragmatic speakers produce *succinct* descriptions (average sentence length 4.21 vs. 4.97) but the pragmatic speakers use a much richer vocabulary (14% more unique nouns and 33% more unique adjectives, after controlling for average length discrepancy).

5.10 Out-of-distribution Transfer Learning

Language is abstract and compositional. These properties make language use generalizable to new situations (e.g. using concrete language in novel scientific domains) and robust to low-level perceptual variation (e.g. lighting). In our final set of experiments we examine the degree to which our neural listeners and speakers learn representations that are correspondingly *robust*: that capture associations between the visual and the linguistic domains that permit generalization out of the training domain.

Understanding out-of-class reference To test the generalization of listeners to novel stimuli, we collected referring expressions in communication contexts made of objects in ShapeNet drawn from new classes: beds, lamps, sofas and tables. These classes are distinct from chairs, but share some parts and properties, making transfer possible for a sufficiently compositional model. For each of these classes we created 200 contexts made of random triplets of objects; and collected 2 referring expressions for each target in each context (from participants on AMT). Examples of visual stimuli and collected utterances are shown in Fig. 5.7 (bottom-row). To this data, we applied an (image-only, with/without-attention) listener trained on the ShapeGlot (i.e. chairs) data. We avoid using point-clouds since unlike VGG which was finetuned with multiple ShapeNet classes, the PC-AE was pre-trained on a single-class.

As shown in Table 5.5, the average accuracy is well above chance in all transfer categories (56% on average). Moreover, constraining the evaluation to utterances that contain *only* words that are in the ShapeGlot training vocabulary (75% of all utterances, column: *known*) only slightly improves the results. This is likely because utterances with unknown words still contain enough known vocabulary for the model to determine meaning. We further dissect the *known* population into utterances that contain part-related words (*with-part*) and their complement (*without-part*). For the training domain of chairs without-part utterances yield slightly higher accuracy. However the useful subcategories that support this performance (e.g. 'recliner') do not support transfer to new categories. Indeed, we observe that for transfer classes the listener performs better when part-related words are present. Furthermore, the performance gap between the two populations appears to become larger as the perceptual distance between the transfer and training domains increases (compare sofas to lamps).

Describing real images Transfer from synthetic data to real data is often difficult for modern machine learning models, that are attuned to subtle statistics of the data. We explored the ability of our models to transfer to real chair images (rather than the training images which were rendered without color or texture from CAD models) by curating a modest-sized (300) collection of chair images from online furniture catalogs. These images were taken from a *similar* view-point to that of the training renderings and have rich color and texture content. We applied the (image-only) *pragmatic* speaker to these images, after subtracting the average ImageNet RGB values (i.e. before passing the images to VGG). Examples of the speaker's productions are shown in Figure 5.7. For each chair, we randomly selected two distractors and asked 2 AMT participants to guess the target given the (highest-scoring) utterance produced by our speaker. Human listeners correctly guessed

Table 5.5: Transfer-learning of neural listeners trained with chair data to novel object classes for different sub-populations of utterances. For reference, the accuracies of the *object generalization* task are included (chairs, first row); The last row reports the average of the transfer/novel categories only. All numbers are *average* accuracies of five listeners trained with different splits of the *object generalization* task (See Section 5.10 for details, and Appendix C for other variants.).

	Population			
Class	entire	known	with part	without part
chair	77.4	77.8	77.0	**80.5**
bed	56.4	55.8	**63.8**	51.5
lamp	50.1	51.9	**60.3**	47.1
sofa	53.6	55.0	**55.1**	54.7
table	63.7	65.5	**68.3**	62.7
average	56.0	57.1	**61.9**	54.9

the target chair 70.1% of the time. Our speaker appears to transfer successfully to real images, which contain color, texture, pose variation, and likely other differences from our training data.

5.11 Conclusion

In this chapter, we have explored models of natural language grounded in the shape of common objects. The geometry and topology of objects can be complex and the language we have for referring to them is correspondingly abstract and compositional. This makes the shape of objects an ideal domain for exploring grounded language learning, while making language an especially intriguing source of evidence for shape variations. We introduced the *ShapeGlot* corpus of highly descriptive referring expressions for shapes in context. Using this data we investigated a variety of neural listener and speaker models, finding that the best variants exhibited strong performance. These models draw on both 2D and 3D object representations and appear to reflect human-like part decomposition, though they were never explicitly trained with object parts. Finally, we found that the learned models are surprisingly robust, transferring to real images and to new classes of objects. Future work will be required to understand the transfer abilities of these models and how this depends on the compositional structure they have learned.

Chapter 6

Referential Language for Object Discrimination in the Real-World

In this final chapter of the thesis we extend the problem of language-driven shape-differentiation presented in the previous chapter (Chapter 5) to a more complex scenario. Specifically, in this chapter we study the problem of learning language-driven object-differentiation for common objects in **real-world 3D scenes**. This is a strictly harder problem, as reference of objects inside real-world scenes, does not involve only shape-based reasoning, but it extends to other object-properties such as their texture or spatial location. Concretely, in this study we focus on a challenging setup where the referred object belongs to a *fine-grained* object class and the underlying scene contains *multiple* object instances of that class. Due to the scarcity and unsuitability of existent 3D-oriented linguistic resources for this task, we first develop two large-scale and complementary visio-linguistic datasets: i) **Sr3D**, which contains 83.5K template-based utterances leveraging *spatial relations* among fine-grained object classes to localize a referred object in a scene, and ii) **Nr3D**, which contains 41.5K *natural, free-form* utterances collected by deploying a 2-player object reference game in 3D scenes. Using utterances of either datasets, human listeners can recognize the referred object with high ($>$86%, 92% resp.) accuracy. By tapping on the introduced data, we develop novel **neural listeners** that can comprehend object-centric natural language and identify the referred object *directly* in a 3D scene. Our key technical contribution is designing an approach for combining linguistic and geometric information (in the form of 3D point clouds similar to those used in Chapter 2) and creating multi-modal 3D neural listeners (similar to those explored in Chapter 5). Crucially, we show that architectures which promote object-to-object communication via graph neural networks outperform less context-aware alternatives, and that fine-grained object classification is a hard bottleneck for

language-assisted 3D object identification.

6.1 Introduction

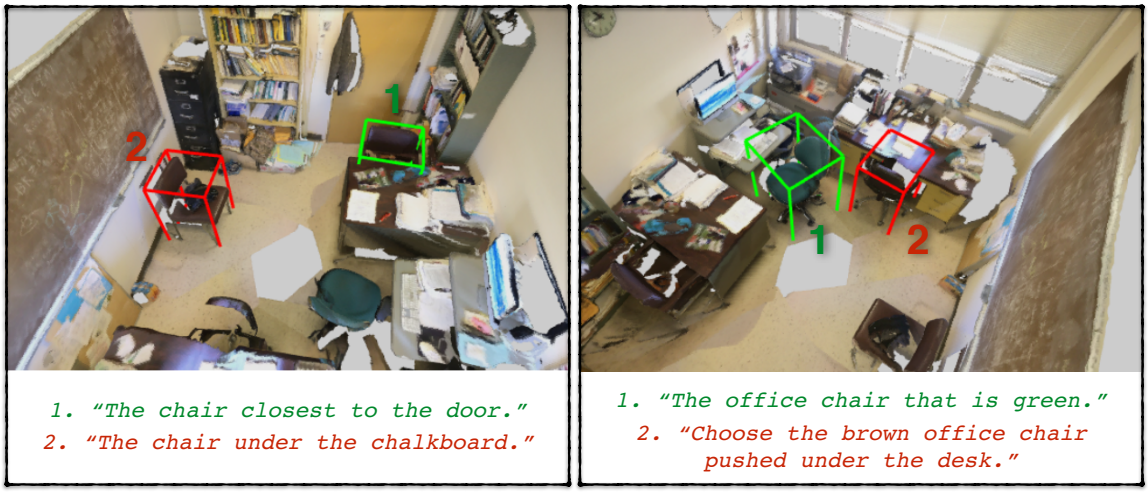


Figure 6.1: **Examples of discriminative utterances and context formation in a 3D scene.** Four chairs in the same scene are contrasted in pairs. The contrasted chairs are of plain type on the left panel, and in the right panel, they are office-type chairs. Each color-coded utterance distinguishes the 'target' object inside the same colored bounding box against a 'distractor' of the **same** fine-grained class. The use of a communication context that includes **all but only** those objects of the same fine-grained type lifts the reference problem **beyond** fine-grained classification.

The progress on connecting language and vision in the past decade has rekindled interest in tasks like visual question answering (e.g., [18, 228]), image captioning (e.g., [123, 278, 302, 178, 10]), and sentence-to-image similarity (e.g., [123, 134]). Recent works have enhanced the accessibility of visual content through language via grounding (e.g., [212, 211]), showing strong results in locating linguistically described visual elements in images. However, most of these works focus on developing better models that connect vision to language in images, which express after all only a 2D view of our 3D reality. Even in embodied AI most works (e.g., embodied QA [65], or embodied visual recognition [303]), fine-grained 3D object identification is not explicitly modeled. Fine-grained 3D understanding however can be essential to more 3D-oriented visually-grounded embodied tasks, such as those that need to be performed by autonomous robotic agents [295, 241]. Representing the 3D shape of objects can also enable the inference and instantiation of geometric constraints for multi-step planning of manipulation of objects in the physical world [208], potentially increasing

the generalization power of robotics in ecological real-world human environments.

Humans possess an astonishing capacity to reason about, describe, and locate 3D objects. Over time we have developed efficient communication protocols to linguistically express such processes – e.g., given the utterance *"the laptop placed on the table next to the main door"*, one can identify the referred object in the room, as long as the reference conveys some unique aspect of that object. Solving such a reference problem *directly* in 3D space – i.e., *without* a camera view dependency – can benefit many downstream robotics applications, including embodied question answering [65], visual- and language-based navigation [14], instruction following [248], and manipulating objects in a scene [297, 154]. Despite this, developing *datasets* and *methods* with characteristics that enable machine learning models to perform well on this 3D reference task is far from straightforward; in this work, we examine how to address both.

Leveraging 3D visual understanding for solving vision and language tasks has been recently explored in Visual Question Answering [142] and Visual Grounding [215]. Still, the focus has been on synthetic datasets without 3D understanding going beyond (at best) multiple 2D views. An alternative, yet more direct way to gain this understanding is by analyzing point cloud data of real-world scenes [6, 222]. Point clouds carry the entire geometric and appearance characteristics of objects and provide access to a larger spatial context (within a scene) than a single 2D view [20]. This flexibility enables us also to bypass camera view dependency (e.g., having access to parts of a scene occluded by a fixed camera) when we refer linguistically to objects.

In this Chapter, we investigate object references when multiple instances of the same fine-grained object class are present in a 3D scene. Discriminative understanding of object classes is important at the fine-grained level and can be achieved with models combining appearance understanding and spatial reasoning skills (e.g., spatial understanding is not critical by itself if we are looking for the unique *office* chair in the presence of one or more *dining* chair(s)). ***Creating discriminative linguistic descriptions:*** We focus on designing a data collection strategy that covers *both* spatial and appearance based identification (Sec. 6.3). As we show in our experiments, this step is critical for progress in 3D visual object identification from free-form language descriptions. Our strategy involves both programmatic and human-based generation of utterances, and has the following characteristics: (i) in every single example there are multiple object instances of the same class referred in the language describing the target 3D object; (ii) in the case of human language utterances, we explicitly ask the human subject to describe a target object in contrast to other instances of the *same* object class. By explicitly contrasting the same fine-grained class instances *and only them,* the resulting utterances are discriminative, even if uttered by crowd-sourced annotators unfamiliar with the environment. Fig. 6.1

illustrates examples of this. ***Developing a 3D neural listener:*** We also design a novel visio-linguistic graph-convolution network that predicts the referred object given a language description, by enabling communication among objects in a 3D visual scene. Our contributions can be summarized below:

1. **Fine-Grained ReferIt3D** task: We introduce the task of language-based identification of specific 3D object instances, where fine-grained object-centric and multi-object understanding is necessary for its completion.

2. **Nr3D** and **Sr3D** datasets: We contribute a new dataset that contains natural and synthetic language descriptions, namely Nr3D and Sr3D respectively. For Sr3D we propose a simple but effective methodology for building template-based and *spatially-oriented* object referential language in 3D scenes. We show that training with Sr3D in addition to natural language data (Nr3D or [54]) improves neural-based pipelines.

3. **ReferIt3DNet**: We explore the task of understanding object references grounded in real-world 3D data (including both language and scenes) by designing a novel visio-linguistic graph neural network, termed *ReferIt3DNet*[*].

6.2 Related Work

2D High-Level Vision & **Language:** Vision & Language, also sometimes called Visual Semantic modeling, has been extensively studied in a variety of 2D tasks. Among early approaches of combining Vision & Semantics are tasks such as zero-shot learning where language/unseen descriptions of an unseen class are provided to describe it (e.g., [304, 150, 11, 232, 254, 166, 165, 270]). Similar approaches have been developed to model image-sentence similarity for bi-directional retrieval of images given a sentence (e.g., [123, 134]). More recently, the development of a large scale dataset of 2D Visual Question answering (VQA) [18] enabled new approaches on how to best represent questions and images for this task. However, a noteworthy dataset-based bias was found: just by inspecting the question/language and without necessarily understanding the visual content, the predictive performance of many models was found to be superficially high [9]. The same bias was shown in tasks such as image-captioning [72]. More balanced VQA benchmarks [91] mitigated some of the biases and motivated the development of better attention mechanisms (e.g., [128]) and modular networks [315, 16]. As per the example set by *2D Vision and Language* community, properly modeling 3D visio-lingual tasks requires establishing carefully designed connections between

[*]The datasets and neural listener code are available at https://referit3d.github.io

language and the 3D visual data, e.g., as these encoded with 3D point clouds.

2D ReferIt Game and Grounded Vision & Language: Several papers explore connecting referential language to image regions for co-reference resolution (e.g., [135, 226] – in videos, e.g., [12]), for generating referring expressions [126, 188, 182], and more. Recent work grounds noun phrases in image captions, such as in Flickr30KEntities [212] and ActivityNetEntities [322] in videos. In [74, 182], the authors proposed the use of referring expressions for human-robot interaction and object localization in real-word environments but using primarily 2D images in contrast to our work.

Visual Relationships and Spatial Reasoning: Detecting visual relationships in images such as `<woman, carrying, umbrella>` (e.g., [168, 2]) has been explored using datasets such as VRD [168] and more recently on the large Visual Genome dataset [138]. Spatial relations have also been studied in 3D by Rosema *et al.* [235]. However, relations in that work are not described in free form and hence are of restricted vocabulary. Also, the goal in [235] is simpler than identifying a target object in a complex 3D environment (our goal).

3D Vision & Language: Connecting 3D vision to natural language is a relatively understudied problem. Within a generative framework, [50] presented conditional generation of 3D models from text, which could be useful in augmented reality applications. In a concurrent work [54], Chen *et al.*, collected natural language to localize referred objects in 3D real-world scenes. Also, in a recent work, Goyal *et al.* collected natural language for minimally contrasting synthetic 3D scenes with a focus on spatial-relations [90]. In contrast to these works, we assume the segmented object instances of a real-world scene and focus on identifying a referred object when other instances with the *same* fine-grained object category exist.

Reference Games. Our 41,503 human language utterances were collected via a reference game played between two humans, as inspired by the 2D, image-oriented ReferItGame [126] and the 3D, shape-oriented, ShapeGlot [7]. The basic arrangement of such games can be traced back to the language games explored by Wittgenstein [289] and Lewis [153]. Recently, these approaches have also been adopted as a benchmark for discriminative and context-aware NLP [203, 15, 193, 58, 276, 259, 148]. Our paper goes beyond this prior work by grounding language behavior in a reference task containing objects in real-world 3D scenes, thereby eliciting rich referential language influenced by the color, the shape, the relative-spatial location, and other properties of the underlying objects.

6.3 Developing Referential 3D-Centric Data

The problem of language driven disambiguation of common objects in real world 3D scenes is new, and as such, not many datasets exist that are well suited for this task. With this in mind we introduce a two-part dataset: a high quality synthetic dataset of referential utterances (**Sr3D**) and a dataset with natural (human) referential utterances (**Nr3D**). Both Sr3D and Nr3D are built on top of ScanNet [63], a real-world 3D scene dataset with extensive semantic annotations that we utilize to create appropriate contrastive *communication contexts*. We define *communication context* as a `(scene, target, distractor(s))` tuple, where *scene* is one of the 707 unique indoor scenes of ScanNet, *target* is one of 76 fine-grained object classes (e.g., office-chairs, armchairs, etc.), and *distractors* are instances of the same fine-grained object class as the target that are contained in the same scene. We generate a total of 5,878 unique tuples. We select the 76 object classes by applying the following intuitive criteria. A class is a valid class for a *target* if: (a) it is contained in at least 5 *scenes*; and (b) each *scene* contains multiple *distractors* but not more than six (to promote a problem beyond fine-grained (FG) classification without making it too hard even for human annotators). We add the constraint of having 5 such *scenes* per class, to foster generalization and make the problem less heavy-tailed (15.26% of all annotated ScanNet classes appear with multiple instances in exactly one *scene*). We also exclude the few classes that are object parts (e.g. a door of a closet) or are structural elements of the scenes (i.e., walls, floors, and ceiling) to ensure that we are working with common objects.

6.3.1 Creating template based spatial references

We introduce the **Spatial Reference in 3D** (**Sr3D**) dataset, consisting of 83,572 utterances. Each utterance aims to uniquely refer to a *target* object in a ScanNet 3D scene by defining a relationship between the *target* and a surrounding object (*anchor*). *Anchors* are object instances that can belong to a set of 100 object classes in ScanNet, comprising of the 76 mentioned above and an additional 24 that: (a) frequently appear as singletons in a scene; and (b) are large objects (e.g., a fireplace or a TV). However, an *anchor* can never belong to the same class as the *target* and, as such, its *distractors*.

Consider, for instance, an underlying 3D scene with a *target* object (e.g. desk) that can be completely disambiguated from its *distractors* with the help of a spatial relation (e.g., closest) to an *anchor* object (e.g., door). We synthesize discriminative **Sr3D** utterances using the following compositional template:

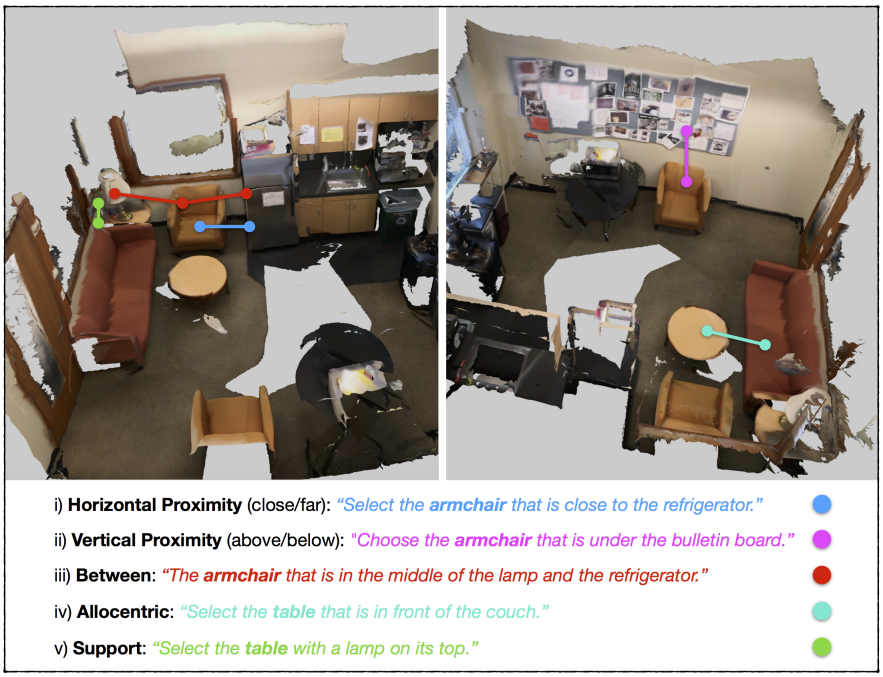


Figure 6.2: **Examples of spatial reference types of Sr3D**. In the left image, there are examples of "horizontal proximity", "between", and "support" relations; the target object in the first two is an armchair and the target in the third relation is a table. In the right image, there are examples of "vertical proximity" and "allocentric" relations; the target objects are an armchair and a table respectively. The left and the right images represent a ScanNet scene where there exist two armchairs (one is beside the refrigerator and the other under the bulletin board) and three tables (a black one under the bulletin board, one in front of the couch, and one in the corner of the room).

$$< \text{target-class} > \quad < \text{spatial-relation} > \quad < \text{anchor-class(es)} > \tag{6.1}$$

e.g., *"the desk that is closest to the door"*. Per (6.1), the **Sr3D** template consists of three placeholders. Our goal is to find combinations of them that can uniquely characterize target objects among their distractors in their scenes.

We define the following five types of spatial object-to-object relations. For more details we refer the reader to Table 6.1 for a summary of statistics and to the Appendix D.

(i) **Horizontal Proximity:** This type indicates how close/far is a *target* from the *anchors* in the scene (Fig. 6.2, i). It applies to distance on the horizontal placement of the objects.

(ii) **Vertical Proximity:** It indicates that the *target* is either above or below the *anchor* (Fig. 6.2,

ii).

(iii) **Between:** Between relations indicate the existence of a *target* between two *anchors* (Fig. 6.2, iii).

(iv) **Allocentric:** Allocentric relations encode information about the location of the *target* with respect to the intrinsic self-orientation of an *anchor* (Fig. 6.2, iv). To define the aforementioned orientation, we need to know; (a) the orientated bounding boxes of the *anchor* and (b) whether the *anchor* has an intrinsic front (e.g., a chair with a back) or not (e.g., a stool). For (a) we utilized the Scan2CAD [23] annotations that provide 9DOF alignments between ShapeNet models and ScanNet objects, and for (b) we used a combination of PartNet's [192] and manual annotations.

(v) **Support:** Support relations indicate that the *target* is either supported by or supporting the *anchor* (Fig. 6.2, v).

Table 6.1: **Statistics of Sr3D.** The first row contains the number of *distinct* communication contexts yielded by each reference-type. The second row contains the number of programmatically generated utterances. *Please note that communication context in the Sr3D setup also takes into account spatial relationships and anchors.*

Relationship	Horizontal Prox.	Vertical Prox.	Support	Allocentric	Between	All
|Context|	34,001	1,589	747	1,880	3,569	41,786
|Utterances|	68,002	3,178	1,494	3,760	7,138	83,572

Discussion Our protocol for generating Sr3D is *simple* but also **effective**:

- A user study conducted in Amazon Mechanical Turk (AMT) revealed that 86.1% of the time, humans guessed correctly the target when provided with a sampled utterance of **Sr3D** (2K samples, $p<0.001$).
- As shown in Sec. 6.5, **Sr3D** allows us to investigate the reference problem in a more controlled manner than **Nr3D**, by providing a homogeneous vocabulary and a specific type of reasoning. For example, it bypasses color- or shape- based reference, and other complicated factual reasoning (e.g., use of brand names or metaphors).

Sr3D+: In addition to the dataset generation described above, we augment Sr3D with more utterances choosing the target object's class among those that do not comply with the criterion of having more than one *distractors* in the scene. Given the synthetic nature of the data, we can generate a large amount of utterances in a cost-free way. We explore the contribution of Sr3D+ to the final performance of our neural listener in Sec. 6.5. This additional set of data will be particularly useful when comparing our method to the *Unique* setting of [54] (Table 6.4), since it assumes that the target object is the only instance of that class in the scene.

6.3.2 Natural reference in 3D scenes

The Natural Reference in 3D (**Nr3D**) dataset contains 41,503 human utterances collected by deploying an online reference game in AMT. The game is played between two humans: a 'speaker' who was asked to describe a designated *target* object in a ScanNet 3D scene and a 'listener' who, given the speaker's utterance, was asked to select the referred object among its *distractors*. The game is structured such that both 'speaker' and 'listener' are rewarded when the *target* is successfully selected, hence incentivising descriptions that are most discriminative in the context of a scene and a general audience.

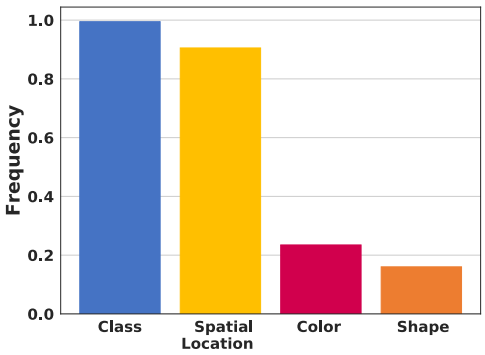


Figure 6.3: Object properties used in sentences of Nr3D (histogram).

Both players are shown the same 3D scene in the form of a decimated mesh model and can interact with it through a 3D interface. In order to remove any camera view bias, we initialize the 'speaker' and 'listener' 3D interfaces with different randomized camera parameters. Given the specifics of the task and the difficulty of understanding the depicted 3D real-world visual content by the non-expert players, we highlight with bounding boxes (oriented when available) the *target* and *distractors*. We distinguish them for the 'speaker' with red and green color respectively, whereas for the 'listener' there is no distinction among them. To encourage players to explore the scene and familiarize themselves with all highlighted objects, we also provide them with a total count of bounding boxes they should expect in the scene. For an example of the speaker's interface we refer the reader to Fig. 1 in Appendix D.

We collect at least 7 utterances from different player pairs per target object. During the collection

process, we iterate over all object instances with the same fine-grained object class in a scene (e.g., all 6 sofa chairs Fig. 1 in Appendix D), providing to the dataset a symmetric property. Among the collected utterances, some originate from games with unsuccessful results; these are not used for training/learning purposes.

Discussion Before presenting our neural agents, we identify several important properties of Nr3D:

- Performance in the gamified data collection process was high (92.2%), but 'listeners' made significantly more errors in the more challenging "hard" contexts (90.0% vs. 94.7%, z = 17.5, p<0.001). We define "hard" contexts as those 3D scenes that contain *more than* 2 *distractors* (Fig. 6.5 illustrates examples of "hard" vs. "easy").

- Speakers naturally produced longer utterances on average to describe targets in hard contexts (approximately 12.5 words vs 10.2, t=-35, p<0.001). The average number of words across all utterances (ignoring punctuation) is 11.4 and the median is 10.

- Regardless of the context difficulty, we identified two attributes in the descriptive power of the utterances (Fig. 6.5): (a) the *target* is **scene-discoverable** when it is uniquely distinguishable among objects in the entire scene and not only its *distractors*. The majority of the utterances mention the fine-grained class type or a close synonym of the *target* (91.6%). This *naturally emerging* property of Nr3D allows us to identify the target among *all* objects in the room; and (b) the identification of the *target* is **view-independent** thus not requiring the observer to place themselves into the scene facing certain objects. Although this attribute is not as prominent as the previous one (63%), even in the case that there is view dependency, speakers were instructed to guide the listeners on how to place themselves in the scene.

- The use of spatial prepositions is ubiquitous (90.5%), which exemplifies why Sr3D is relevant. Reference to color and shape properties is drastically less used in distinguishing instances of the same fine-grained object class. Fig. 6.3 shows a histogram reflecting how often different types of object properties are used accross all sentences of Nr3D.

6.4 Developing 3D Neural Listeners

Given a 3D scene $\mathcal{S}$ represented as an RGB-colored point-cloud of N points $\mathcal{S} \in \mathbb{R}^{N\times 6}$ and a word-tokenized utterance $U = \{u_1, \ldots, u_t\}$ we want to build a neural listener that can identify the

referred target object $\mathcal{T} \subset \mathcal{S}$. To this end, we assume access to a partition of $\mathcal{S} = \{O_1, \ldots, O_M\}$ that represents the objects (O_i) present in $\mathcal{S}$. While it is feasible to attempt identification (or more precisely, in this case, object localization) by operating directly on the unstructured $\mathcal{S}$ ([218], [54]), the problem of instance localization (*especially* for FG classes) remains vastly unsolved. To overcome this and decouple the 3D instance-segmentation problem from our referential setting, we assume access to the instance-level object segmentations of the underlying scene. This choice allows us to cast the 3D reference problem into a classification problem that aims to predict the referred "target" among M segmented 3D instances.

While the above assumption eliminates the need to define each object in $\mathcal{S}$, it still leaves open the problems of: (i) FG object classification; (ii) recognition of the referred object class (per the

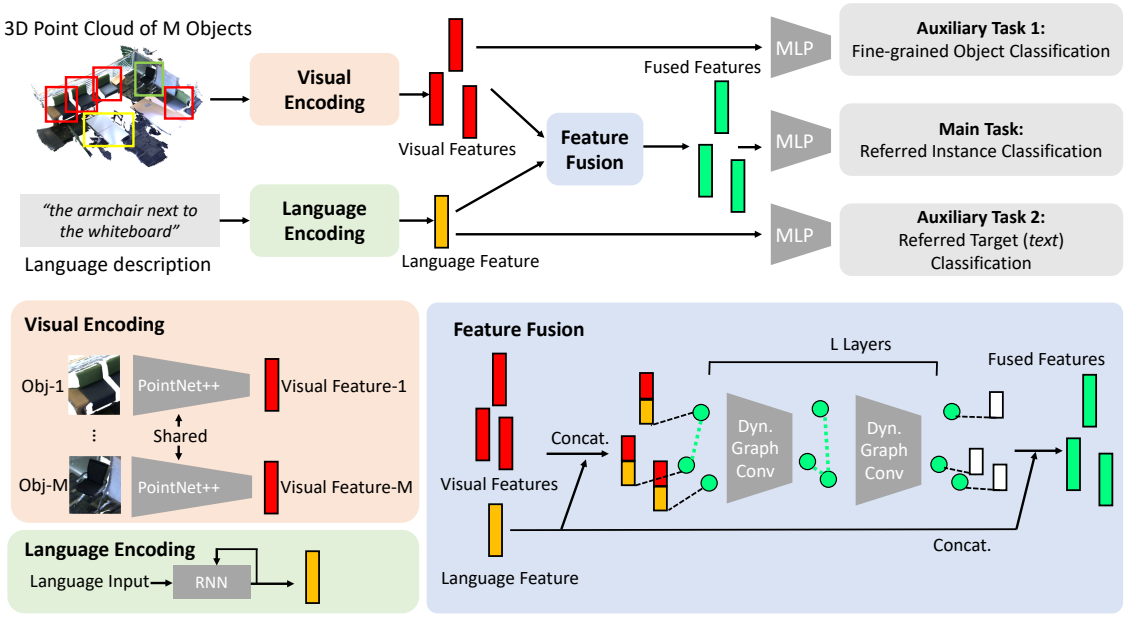


Figure 6.4: **The *ReferIt3DNet* neural listener**. Each object of a 3D scene, represented as a 6D point cloud containing its xyz coordinates and RGB color, is encoded by a visual encoder (e.g., PointNet++), with shared weights. Simultaneously, the utterance describing the referred object (e.g., "*the armchair next to the whiteboard*") is processed by a Recurrent Neural Network (RNN). The resulting representations are fused together and processed by a Dynamic Graph Convolution Network (DGCN) which creates an object-centric *and* scene- (context-) aware representation per object. The output of the DGCN is processed by an MLP classifier that estimates the likelihood of each object to be the referred one. Two auxiliary losses modulate the unfused representations before these are processed by the DGCN via an FG object-class classifier and a referential-text classifier respectively ($\mathcal{L}_{fg}$ and $\mathcal{L}_{text}$ – see text for details).

utterance); and (iii) the original problem of selecting the referred object among the m options. For the first two tasks, we experiment with a neural listener that utilizes two auxiliary cross-entropy losses ($\mathcal{L}_{fg}$, $\mathcal{L}_{text}$) aimed to decouple these intrinsic aspects of the original task. Specifically, the two losses are added to the cross-entropy loss of the main task in hand ($\mathcal{L}_{ref}$) making a final loss that is a weighted sum of these terms:

$$\mathcal{L}_{total} = \alpha_1 \mathcal{L}_{fg} + \alpha_2 \mathcal{L}_{text} + \mathcal{L}_{ref}$$

Contextual scene understanding The above design is object-aware, but our underlying task is also scene-oriented and heavily relies on the *configuration* of the objects present in a scene. Because of this reason it is important to provide a neural listener with a signal that contains explicit information about the scene it operates. A baseline that we explored to this end, is to create a PointNet++ hierarchical scene-feature (based on a large number of points of $\mathcal{S}$) which we fused with every visual representation extracted independently for each object O_i. While the resulting representation is simultaneously object-centric and scene-aware it is not taking into account explicit object-to-object interactions. A more sophisticated approach – which is part of the **ReferIt3DNet** – uses a structured and explicit way of capturing object-to-object interactions to provide information about the scene. Specifically, we use a dynamic graph-convolutional network (DGCN) [285] that operates on the visual features of the objects present in a scene (the object are nodes of a graph). The edges of this graph are computed dynamically at each layer of the DGCN according to the Euclidean similarity among the updated (per-node) visual features. In our experiments we use the k-nearest neighbor-graph among the nodes ($k = 7$, chosen per validation). We note that $k = 7$ creates a relatively sparse graph (the 90^{th} percentile of the number of objects in the training scenes is 52). For further details we refer the reader to the Appendix D.

Incorporation of language An important decision regards how one should "fuse" the linguistic signal in a pipeline like the above. Despite a chair being visually different from a door, our graph-network should inspect the relation among these objects, especially when the reference requires it (e.g., "the chair close to the door"). To promote this action we fuse the visual (object) features with the linguistic ones (derived by an RNN) *before* we pass them to the DGCN. We also explore the effect of adding the linguistic features *after* the DGCN and in both places – which is the best performing option. An overview of our pipeline is illustrated in Fig. 6.4.

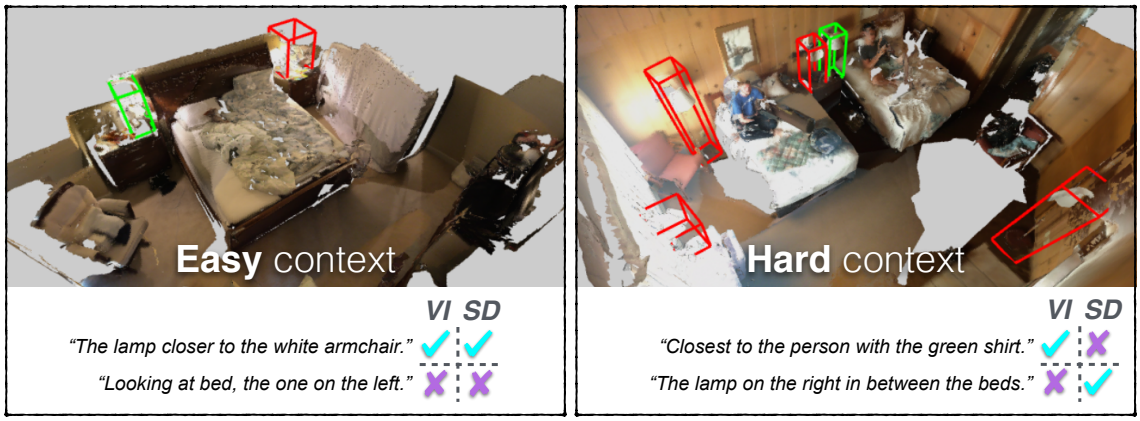


Figure 6.5: **Key properties of referential language and contexts in 3D environments.** First, is the reference being made within a communication context that is "Easy" containing only a few distractors (left image) or "Hard" (right image)? Second, is the utterance permitting a navigating agent to discover the target among *all* objects of the scene, making it **Scene-Discoverable (SD)**? Note, that by construction of the communication contexts of Nr3D, if the utterance can disambiguate among the fine-grained distractors, and also mentions the target's object class (or a synonym) it is SD. Third, does the reference rely on the listening agent to consider a specific/narrow view-point of the scene (e.g., *"Looking at (the front of) the bed..."*), or the description makes the target identifiable from several views, i.e., is View-Independent (VI)?

6.5 Experiments & Analysis

We explore different listening architectures [†] and report the listening accuracy; each test utterance receives a binary score (1 if the correct object is predicted as target and 0 otherwise). For all experiments we use the official-ScanNet splits.

1. **Decoupled approach**: This is a baseline listener consisting of a text classifier and an (FG) object classifier that are trained separately. Given an utterance we use the text classifier to predict the referred object-class. Then we select uniformly i.i.d. (and output) an object from $O_i \in \mathcal{S}$ for which the object classifier matches the text-based prediction. We note that in Nr3D (Sr3D) test accuracies for the two classifiers are 93.0% (100.0%) and 64.7% (67.4%), indicating a noticeable asymmetry in the difficulty of solving the two tasks.

2. **Vision + Language, no Context (V + L)**: Inspired by context-free listening architectures like those in [7, 193], we ground an RNN with the visual feature of each object of $O_i \in \mathcal{S}$, *independently*, and use a shallow classifier to predict the likelihood of each O_i for being the

[†] Architecture details and hyper-parameters for all the experiments, are provided in the Appendix D.

Table 6.2: **ReferIt3DNet performance on Nr3D with/out Sr3D**. The first row contains the achieved accuracy on the Nr3D testing data for a listener trained solely with the Nr3D training set; the other rows showcase the effect of training simultaneously with the Sr3D/Sr3D+, respectively.

	Overall	Easy	Hard	View-dep.	View-indep.
Nr3D	35.6%±0.7%	43.6%±0.8%	27.9%±0.7%	32.5%±0.7%	37.1%±0.8%
w/ **Sr3D**	37.2%±0.3%	44.0%±0.6%	**30.6%±0.3%**	**33.3%±0.6%**	39.1%±0.2%
w/ **Sr3D+**	**37.6%±0.4%**	**45.4%±0.6%**	30.0%±0.4%	33.1%±0.5%	**39.8%±0.4%**

referred target. This baseline can encode visual properties of an object beyond its FG class enabling rich (context-free) distinctions (e.g.,"very small, or yellow colored chair").

3. **Vision + Language + Holistic Context (V + L + C)**: Similar to the above, but also fuses a PointNet++ scene-feature with each object's visual feature to ground the RNN. This enables the inspection of non-structured context when solving the reference task (PointNet++ is applied on a non-segmented scene point cloud).

4. **Vision + Language + Graph (structured) Context (*ReferIt3DNet*)**: This is our proposed listener and comes in three variants that differ w.r.t. *where* we fuse the linguistic with the visual information.

Neural Listeners. Comparisons for the above models are presented in Table 6.3. We observe the following main trends[‡]: i) using the visual and linguistic auxiliary classification losses improves performance; ii) Simplified language (Sr3D) makes identification easier; iii) scene context matters a lot, but most importantly how we incorporate the context (e.g., via DGCN, or direct fusion of PointNet++) makes an important difference in performance. As expected, a more structured versus a rudimentary representation favors better results; iv) where we fuse language matters as well: ReferIt3DNet-A fuses after the DGCN, ReferIt3DNet-B before, and the best performing (for Nr3D) model fuses in both places.

The results shown in Fig. 6.6 show the neural listener's capability to understand and locate objects in challenging 3D scenarios. For example, the top-right example was successful despite the utterance being long. Referring to this particular trashcan among other similar ones requires both spatial reasoning and visual 3D understanding. Similar capability can be demonstrated in the two examples in the second row about the door and the cabinets. Finally, the last row shows two challenging failure cases of our model. In the bottom-left example, the utterance has wrongly placed

[‡]In all results mean accuracies and standard errors across 5 random seeds are reported, to control for the point cloud scene sampling.

Table 6.3: **Listening performance of various ablated models.** The first two columns contain the obtained accuracy when no auxiliary losses are used, and the last two the accuracy when these losses are included.

	Nr3D	**Sr3D**	**Nr3D**	**Sr3D**
Aux. classification loss	**No**		**Yes**	
Decoupled	25.5%	31.7%	-	-
V + L	26.1%±0.5%	32.6%±0.4%	26.6%±0.5%	33.0%±0.4%
V + L + C	27.5%±0.6%	34.7%±0.4%	28.5%±0.6%	37.2%±0.4%
ReferIt3DNet-A	32.3%±0.3%	**39.7%±0.3%**	33.4%±0.3%	**41.0%±0.3%**
ReferIt3DNet-B	31.8%±0.3%	38.1%±0.2%	33.0%±0.3%	40.5%±0.2%
ReferIt3DNet	**32.4%±0.5%**	38.4%±0.2%	**35.6%±0.7%**	39.8%±0.2%

Table 6.4: **ScanRefer performance with/out Sr3D.** MeanIoU improvements when combining Sr3D data with ScanRefer's data during training.

Dataset	**Unique**		**Multiple**		**Overall**	
	P@0.25	**P@0.5**	**P@0.25**	**P@0.5**	**P@0.25**	**P@0.5**
ScanRefer	53.8%	37.5%	21.0%	12.8%	26.4%	16.9%
w/ **Sr3D**	60.0%	39.1%	21.7%	14.3%	28.0%	18.4%
w/ **Sr3D+**	**63.6%**	**42.2%**	**24.1%**	**15.8%**	**30.6%**	**20.1%**

the listener in the room (the chair is at the 3 o'clock position instead of 9). The bottom-right example is particularly hard to solve; the scene is almost symmetric and stripped of visual features, making it hard to discriminate among the chairs.

Combining Nr3D & Sr3D. In Table 6.2, we observe how combining the two datasets provides a consistent boost in performance. This demonstrates the benefit of training with a synthetically generated dataset in addition to a dataset containing natural utterances. We see a similar outcome when combining Sr3D with the ScanRefer [54] training data (see Table 6.4). Specifically, for this experiment we used the publicly available author's implementation of the "xyz+rgb+lobjcls" localization variant ([53]), which is most similar in terms of the used input to our setup. We also note that the results in Table 6.4 concerning training without the addition of Sr3D (first row) are taken from ScanRefer's [v1] arXiv report which was available at the time of writing this paper. Going back to Table 6.2, the results showcase that our neural listener performs better by a large margin in "easy" versus "hard" cases. This is expected and highlights that more work needs to be done in distinguishing objects when there are many fine-grained-same-class distractors in a scene. Another interesting finding is that view-independent utterances are easier to comprehend than view-dependent ones. This is an intuitive finding since the network naturally has more work to do to understand

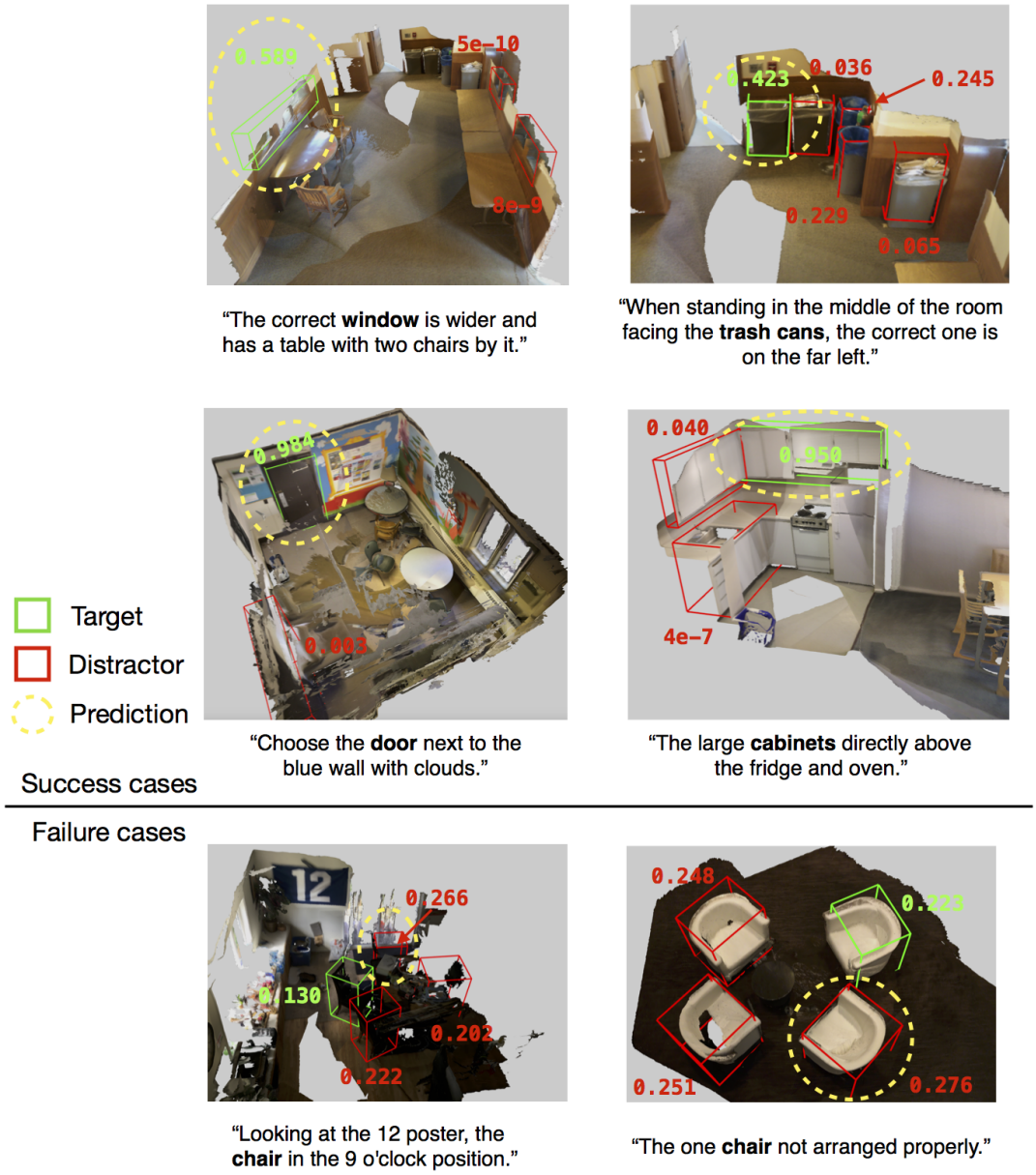


Figure 6.6: **Qualitative results**. Successful cases of applying *ReferIt3DNet* are shown in the top four images and failure ones in the bottom two. Targets are shown in green boxes, intra-class distractors in red, and the referential text is displayed under each image. The network predictions are shown inside dashed yellow circles, along with the inferred probabilities. We omit the probabilities of inter-class distractors to ease the presentation.

nuances related to the view of the scene relative to a specific object.

6.6 Conclusion

Language assisted object disambiguation done directly for 3D objects in 3D environments is a novel but very challenging task. This is especially true when one tries to distinguish among multiple instances of the same fine-grained object category. In addition to the intrinsic difficulty of the problem, there is a scarcity of appropriate datasets. Creating relevant visio-linguistic data that allow us to study this problem is important for advancing 3D deep-learning that, similar to 2D visual learning, is a data hungry methodology. While our neural listeners are a promising first step, more research has to be done before human-level performance and generalization is attained. In summary, this chapter has (a) introduced the problem of fine-grained multi-instance 3D object identification in real-world scenes; (b) contributed two relevant public datasets; and (c) explored an array of sensible neural architectures for solving the referential task.

Chapter 7

Conclusions & Future Work

7.1 Connecting the Dots

This thesis systematically introduced our work on 3D object-centric machine learning by focusing on it from two complementary angles: introducing generative networks for 3D shape **synthesis** and language-guided discriminative networks for 3D object/shape **differentiation**. Our works, taken together, have explored deep-learning-based methods involving most of the major digital formats for representing 3D objects: point clouds, voxel grids, meshes, and 2D object renderings – adapting the techniques to the intricacies of each modality, but also abstracting away from it to provide a general treatment of 3D objects and learning. We begun in Chapter 2 by introducing self-supervised neural models that can learn semantically rich latent representations of 3D objects represented as point clouds. The resulting latent space while being powerful in several perception tasks, e.g., improving the state-of-the-art in unsupervised object recognition, did not offer a straightforward mechanism to manipulate the latent representation, aside from doing 'basic' linear algebra. The next Chapter 3, addressed partially this limitation by introducing an intuitive structural bias in the construction of the latent space: the latents should obey a factorization w.r.t. the commonly found part-based composition of 3D objects. Tapping on this factorization we introduced novel applications for deep shape synthesis including part-aware latent interpolations of objects, and mixing-&-matching their parts also in the latent space. The next chapter, Chapter 4, also concerns the creation of a latent space for 3D objects, but unlike the previous two it does not apply an auto-encoding like scheme. Instead, its main idea is the proposition of learning to reconstruct a shape from a compact matrix that encapsulates how a shape is geometrically different from a fixed 'base' shape. Tapping on the matrix nature of its inputs this approach results in a latent space where multiplicative (matrix) algebra is

possible which in turn often lead in better shape interpolations and analogies compared to the linear approach of Chapter 2.

Overall, these first three chapters concern generative networks and latent space constructs for 3D shapes with various trade-offs. However, a key desideratum that *any* latent space of shapes should obey, is to be able to represent all the diversity of the underlying distribution of shapes. Developing techniques that can ensure such a property, naturally implies the development of trustworthy methods for **measuring how (any) two shapes are different**. While Chapter 4 provided some initial tools in this direction, e.g., learning how two shapes are geometrically different given dense correspondences, humans typically approach this problem differently. Humans can abstract a shape w.r.t. to its parts, style, average size (w.r.t. an imaginative shape collection), etc. – and by using natural language can succinctly describe a shape's most salient differences against any other shape. Chapter 5 is built upon these intuitions and provided an in depth study that connected shape-differentiation and natural language. It introduced a relevant dataset (ShapeGlot) and the first neural-listeners/speakers that similar to humans can reason about pragmatic shape differences with language. Interestingly, the developed neural systems acquired a strong sense of shape part-awareness *only* under the guiding training signal of referential language – i.e., without operating in latent spaces that were explicitly trained to capture the part-based composition of objects (like that of Chapter 3). Finally, with our last chapter, Chapter 6 we extended the scope of our investigation on the interplay between differentiating language and 3D shapes. Unlike the previous chapters, here, we focused on how an object might be different from other objects of the same class, but with which it *co-exists inside a real-world scene*. Similarly to Chapters 5 and 6 we used point clouds to represent the objects' geometries and textures, and language to form the differentiating signal. However, for this cognitive task, as our introduced visio-linguistic dataset (ReferIt3D) revealed, while shape and texture are frequently used by humans to differentiate an object, its non-unary relation to other objects, e.g., its spatial location, is almost always necessary in identifying in a scene. This fact in turn demanded the creation of a more complicated latent space for objects than the previous ones explored in this thesis – one where objects were analyzed *jointly* (via graph-neural-networks) to reflect how they co-exist in a scene.

7.2 Going Forward

This thesis covers a relatively broad spectrum of research material, as it attempts to bring closer together two distinct topics: the generation of 3D objects and referential language concerning them. Moreover, each of its primary works (Chapter 2, Chapter 5, Chapter 6), introduced **a new**

problem in the deep learning literature at the time of the corresponding publication. Because of these conditions, naturally there are still many relevant problems, techniques and ideas that could be used in conjunction with the presented material to further explore how one can deep learn referential language and 3D objects together. After all, this is a very recent (but rapidly evolving) direction in the Computer Vision and Computational Linguistics communities. Below, I present some of these untapped ideas and problems that I find promising and which I hope will be explored in more depth in the near future. First, I will describe some straightforward ways one could try to connect further the "dots" of this thesis, by exploiting together and in novel ways some of the presented materials. Then, I will present a brief open-ended discussion of closely related problems in the broader context of connecting language to vision.

7.2.1 Dots left disconnected

The two most salient possibilities that one could exploit to further bridge the gap between shape and language by using materials from this thesis, concern the incorporation of ideas from Chapters 3 and 4 in the creation of multi-modal neural-listeners/speakers like those presented in the later chapters. Specifically, and as it is discussed above, the developed multi-modal neural-nets of Chapter 5 acquire an understanding of the semantic parts an underlying shape has, from language alone, without ever using them explicitly in their respective loss functions. However, by using existing collections of semantic 3D shape parts like PartNet [192], and ideas similar to those presented in Chapter 3, one could design neural-listeners/speakers that embed the underlying objects into a latent space that factorizes their semantic parts instead of using a holistic latent space like the one presented in Chapter 2. Such an action should promote a disentangled treatment among the referred parts-words of a sentence ('the leg', 'the back' etc.) and their actual underlying 3D geometries (e.g., the point cloud corresponding to a chair's leg or back) . As we already have shown, a simple *word-to-word* attention mechanism can automatically learn to focus on part words when relevant for the reference task (Figure 5.4). It is conceivable that we can extend this mechanism to a *word-to-part* attention, that learns the mapping among part words and their corresponding 3D semantic parts; which in turn can be used to *localize* where the neural reasoning takes place, i.e., attend to the latents of the mentioned part(s).

Aside of the ideas of Chapter 3, the main proposal of Chapter 4 also offers some appealing potential for improving the quality and performance of multi-modal neural-listeners/speakers that jointly examine geometry and discriminative language. Chapter 3 explores the usage of shape-difference matrices that capture how one shape is different shape-wise from another and attempts

deep learning them. First, it would be interesting to see to what extent one can use such matrices in place of the latent representations arising from auto-encoding-like schemes. Of course, given the requirement of having correspondences to create shape-difference matrices; one would have to rely here on extracting approximate correspondences (e.g., [201]) to work with arbitrary shapes. Second, one can extract shape-differences among and only the shapes that are being contrasted (e.g, triplets of ShapeGlot or scene-objects of ReferIt3D), instead of using an a priori fixed base shape which should enhance the relevance of such a signal for the down-stream reference task. Last, it would be interesting to try the reverse direction: using referential (linguistic) data to influence the creation of geometry-aware shape-differences; since referential language tends to focus on the most salient shape/object differences it can provide a focal point for where most geometric comparisons underlying a shape difference matrix should occur.

7.2.2 Discussion of open relevant problems and limitations

Deep learning is clearly taking our world by storm and the future opportunities in applying it seem vast. This thesis concerns multi-modal learning more than anything else, which is still a relatively underdeveloped field. A key difficulty when it comes to connecting language to vision (as in 3D objects) concerns the difference these two modalities have in terms of their generality as discussed in Section 1.1.4. How to equip a latent visual space to better reflect the richness/details that one can express in language about the underlying stimuli is not exactly known. One can think of creating latent-spaces for visual objects by *jointly* inspecting the object collection **and** external language that describes differences or properties of the encoded objects. By having access to the rich-label set of language one can at least hope to pay more attention to the language's most salient/common-found expressed semantics, while they build visual generative models. Assuming that the collected human language talks about the most important aspects of the encoded items, paying more attention to it (with a multi-modal loss function), should create latent spaces that better preserve the crucial visual details. However, any such an approach, does not appear to be scalable, as collecting referential language that is ''complete" in the sense that it captures all essential visual properties in all possible comparisons is clearly infeasible. For simple shapes of a limited class of common objects we created ShapeGlot. For contrasting monochromatic colors, *Monroe et al.* [193] collected another large scale dataset. How do we escape such highly specialized (and expensive) datasets to scale referential language *in the wild*? It seems reasonable that we will need some smart prior that can promote transferability among the learned representations and tasks. For instance, an extreme prior is to assume a product-like factorized decomposition of reference-making in the wild, that combines

measurements about distinct properties of the visual world, like its color and shapes, independently. Such a prior, which seems too optimistic, would drastically reduce the language-acquisition cost and the underlying learning complexity. In general, it would be exciting to see new research that brings somewhat established ideas and priors, described in cognitive science concerning the vision/language world (e.g., [268, 70, 44]), in the context of learning with neural networks.

Finally, a few practical problems and applications that are low-hanging fruits directly related to this thesis are the following: First, can we use referential language as an auxiliary signal to enable directed **changes** over a given stimulus – say make the legs of given chair thinner. Our recent paper on shape deformation learning [261] finds latent directions that one can use to change (consistently) the shape of objects in a collection. Language could be a used here as a user-friendly "knob" that expresses the actual change a user wants to perform w.r.t. to a specific input item. Similarly and secondly, one can extend language for **manipulating** objects embedded in 3D scenes: for that we would need machines like the neural-listeners presented in Chapter 6 which can identify and isolate the object(s) we want to manipulate; and mechanisms that can act on them, e.g., to change their location, appearance etc. ([297, 172]). Last, creating discriminative **neural-speakers** for objects embedded in 3D scenes, is a promising application that can find usages in making smarter 3D interactive environments for mixed/augmented reality. Given the performance gap between humans and our 3D neural-listeners from Chapter 6, attempting to implement their (harder) generative speaking counterparts, like we did in Chapter 5, seems too bold. Luckily, the last few months since our introduction of the 3D-listening problem [4] a spark of interest has been ignited with many new approaches improving the state-of-the-art [78, 306, 209, 231, 100, 316], increasing the range of possible novel applications.

Bibliography

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, 2016.

[2] Sherif Abdelkarim, Aniket Agarwal, **Panos Achlioptas**, Jun Chen, Jiaji Huang, Boyang Li, Kenneth Church, and Mohamed Elhoseiny. Long tail visual relationship recognition with hubless regularized relmix. In *International Conference on Computer Vision (ICCV)*, 2021.

[3] **Panos Achlioptas**. Stochastic gradient descent in theory and practice, 2016. Stanford M.Sc. Thesis.

[4] **Panos Achlioptas**, Ahmed Abdelreheem, Fei Xia, Mohamed Elhoseiny, and Leonidas J. Guibas. ReferIt3D: Neural listeners for fine-grained 3d object identification in real-world scenes. In *European Conference on Computer Vision (ECCV)*, 2020.

[5] **Panos Achlioptas**, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J. Guibas. Representation learning and adversarial generation of 3D point clouds. *Workshop on Implicit Models, International Conference of Machine Learning (ICML)*, 2017.

[6] **Panos Achlioptas**, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J. Guibas. Learning representations and generative models for 3D point clouds. In *International Conference on Machine Learning (ICML)*, 2018.

[7] **Panos Achlioptas**, Judy Fan, Robert XD Hawkins, Noah D Goodman, and Leonidas J. Guibas. ShapeGlot: Learning language for shape differentiation. In *International Conference on Computer Vision (ICCV)*, 2019.

[8] **Panos Achlioptas**, Maks Ovsjanikov, Kilichbek Haydarov, Mohamed Elhoseiny, and Leonidas Guibas. ArtEmis: Affective language for visual art. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[9] Aishwarya Agrawal, Dhruv Batra, and Devi Parikh. Analyzing the behavior of visual question answering models. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

[10] Harsh Agrawal, Karan Desai, Yufei Wang, Xinlei Chen, Rishabh Jain, Mark Johnson, Dhruv Batra, Devi Parikh, Stefan Lee, and Peter Anderson. nocaps: novel object captioning at scale. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[11] Zeynep Akata, Scott Reed, Daniel Walter, Honglak Lee, and Bernt Schiele. Evaluation of output embeddings for fine-grained image classification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[12] Hazan Anayurt, Sezai Artun Ozyegin, Ulfet Cetin, Utku Aktas, and Sinan Kalkan. Searching for ambiguous objects in videos using relational referring expressions. *Computing Research Repository (CoRR)*, abs/1908.01189, 2019.

[13] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[14] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[15] Jacob Andreas and Dan Klein. Reasoning about pragmatics with neural listeners and speakers. *Computing Research Repository (CoRR)*, abs/1604.00562, 2016.

[16] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[17] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. SCAPE: Shape Completion and Animation of People. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 408–416. ACM, 2005.

[18] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *International Conference on Computer Vision (ICCV)*, 2015.

[19] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *Computing Research Repository (CoRR)*, abs/1701.07875, 2017.

[20] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3D semantic parsing of large-scale indoor spaces. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[21] Sanjeev Arora and Yi Zhang. Do gans actually learn the distribution? an empirical study. *Computing Research Repository (CoRR)*, abs/1706.08224, 2017.

[22] K Somani Arun, Thomas S Huang, and Steven D Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 1987.

[23] Armen Avetisyan, Manuel Dahnert, Angela Dai, Manolis Savva, Angel X. Chang, and Matthias Nießner. Scan2CAD: Learning CAD model alignment in RGB-D scans. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[24] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *Computing Research Repository (CoRR)*, abs/1607.06450, 2016.

[25] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *Computing Research Repository (CoRR)*, abs/1409.0473, 2014.

[26] Jeremy Barnes, Roman Klinger, and Sabine Schulte im Walde. Projecting embeddings for domain adaption: Joint modeling of sentiment analysis in diverse domains. *Computing Research Repository (CoRR)*, abs/1806.04381, 2018.

[27] David Bau, Alex Andonian, Audrey Cui, YeonHwan Park, Ali Jahanian, Aude Oliva, and Antonio Torralba. Paint by word. *Computing Research Repository (CoRR)*, abs/2103.10951, 2021.

[28] Heli Ben-Hamu, Haggai Maron, Itay Kezurer, Gal Avineri, and Yaron Lipman. Multi-chart generative surface modeling. In *Proc. SIGGRAPH Asia*, page 215. ACM, 2018.

[29] Heli Ben-Hamu, Haggai Maron, Itay Kezurer, Gal Avineri, and Yaron Lipman. Multi-chart generative surface modeling. *ACM Trans. Graph.*, 37(6):215:1–215:15, December 2018.

[30] Yoshua Bengio, Eric Laufer, Guillaume Alain, and Jason Yosinski. Deep generative stochastic networks trainable by backprop. In *International Conference on Machine Learning (ICML)*, 2014.

[31] Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

[32] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[33] Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Dynamic FAUST: Registering human bodies in motion. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[34] Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Dynamic FAUST: Registering human bodies in motion. In *CVPR*, July 2017.

[35] Davide Boscaini, Davide Eynard, Drosos Kourounis, and Michael M Bronstein. Shape-from-operator: Recovering shapes from intrinsic operators. In *Computer Graphics Forum*, 2015.

[36] Davide Boscaini, Jonathan Masci, Emanuele Rodolà, and Michael Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[37] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *Computing Research Repository (CoRR)*, abs/1511.06349, 2015.

[38] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2001.

[39] André Brock, Theodore Lim, James M. Ritchie, and Nick Weston. Generative and discriminative voxel modeling with convolutional neural networks. *Computing Research Repository (CoRR)*, abs/1608.04236, 2016.

[40] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. *Computing Research Repository (CoRR)*, abs/1809.11096, 2019.

[41] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 2017.

[42] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *Computing Research Repository (CoRR)*, abs/1312.6203, 2013.

[43] Kenneth R. Castleman. *Digital Image Processing*. Prentice Hall Press, 1996.

[44] P. Cavanagh. The language of vision. *Perception*, 2003.

[45] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Nießner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from rgb-d data in indoor environments. *Computing Research Repository (CoRR)*, abs/1709.06158, 2017.

[46] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An information-rich 3D model repository. *Computing Research Repository (CoRR)*, abs/1512.03012, 2015.

[47] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, 2006.

[48] Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode regularized generative adversarial networks. *Computing Research Repository (CoRR)*, abs/1612.02136, 2016.

[49] Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. On visual similarity based 3d model retrieval. *Computer Graphics Forum*, 2003.

[50] Kevin Chen, Christopher B Choy, Manolis Savva, Angel X Chang, Thomas Funkhouser, and Silvio Savarese. Text2shape: Generating shapes from natural language by learning joint embeddings. *Computing Research Repository (CoRR)*, abs/1803.08495, 2018.

[51] Wenzheng Chen, Huan Ling, Jun Gao, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3D objects with an interpolation-based differentiable renderer.

In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[52] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[53] Z. Dave Chen, Angel X. Chang, and Matthias Nießner. https://github.com/daveredrum/ScanRefer. Accessed: 2020-07-17.

[54] Z. Dave Chen, Angel X. Chang, and Matthias Nießner. ScanRefer: 3D object localization in RGB-D scans using natural language. *Computing Research Repository (CoRR)*, abs/1912.08830, 2019.

[55] Zhiqin Chen, Kangxue Yin, Matthew Fisher, Siddhartha Chaudhuri, and Hao Zhang. BAE-NET: Branched autoencoder for shape co-segmentation. *Computing Research Repository (CoRR)*, abs/1903.11228, 2019.

[56] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. *Computing Research Repository (CoRR)*, abs/1812.02822, 2018.

[57] Herbert H Clark and Deanna Wilkes-Gibbs. Referring as a collaborative process. *Cognition*, 22(1):1–39, 1986.

[58] Reuben Cohn-Gordon, Noah Goodman, and Christopher Potts. Pragmatically informative image captioning with character-level inference. *Computing Research Repository (CoRR)*, abs/1804.05417, 2018.

[59] Lia Coleman, **Panos Achlioptas**, and Mohamed Elhoseiny. Towards a principled evaluation of likeability for machine-generated art. In *Machine Learning for Creativity and Design workshop on Neural Information Processing Systems (NeurIPS)*, 2019.

[60] Etienne Corman. *Functional representation of deformable surfaces for geometry processing*. PhD thesis, Université Paris-Saclay, 2016.

[61] Etienne Corman, Justin Solomon, Mirela Ben-Chen, Leonidas Guibas, and Maks Ovsjanikov. Functional characterization of intrinsic and extrinsic geometry. *ACM Transactions on Graphics (TOG)*, 2017.

[62] Trevor F. Cox and M.A.A. Cox. *Multidimensional Scaling, Second Edition*. Chapman and Hall/CRC, 2 edition, 2000.

[63] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[64] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor CNNs and shape synthesis. *Computing Research Repository (CoRR)*, abs/1612.00101, 2016.

[65] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[66] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[67] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 1977.

[68] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[69] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[70] B. Dessalegn and B. Landau. Interaction between language and vision: it's momentary, abstract, and it develops. *Cognition*, 2013.

[71] Jaime Deverall. Using generative adversarial networks to design shoes : The preliminary steps, 2017.

[72] Jacob Devlin, Saurabh Gupta, Ross Girshick, Margaret Mitchell, and C Lawrence Zitnick. Exploring nearest neighbor approaches for image captioning. *Computing Research Repository (CoRR)*, abs/1505.04467, 2015.

[73] Nat Dilokthanakul, Pedro AM Mediano, Marta Garnelo, Matthew CH Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoders. *Computing Research Repository (CoRR)*, abs/1611.02648, 2016.

[74] Fethiye Irmak Doğan, Sinan Kalkan, and Iolanda Leite. Learning to generate unambiguous spatial referring expressions for real-world environments. *Computing Research Repository (CoRR)*, 2019.

[75] Anastasia Dubrovina, Fei Xia, **Panos Achlioptas**, Mira Shalah, Raphael Groscot, and Guibas Leonidas J. Composite shape modeling via latent space factorization. *International Conference on Computer Vision (ICCV)*, 2019.

[76] Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A point set generation network for 3d object reconstruction from a single image. *Computing Research Repository (CoRR)*, abs/1612.00603, 2016.

[77] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2010.

[78] Mingtao Feng, Zhen Li, Qi Li, Liang Zhang, Xiangdong Zhang, Guangming Zhu, Hui Zhang, Yaonan Wang, and Ajmal Mian. Free-form description guided 3D visual graph network for object grounding in point cloud. *Computing Research Repository (CoRR)*, abs/2103.16381, 2021.

[79] A. Martin Fischler and Elschlager A. Robert. The representation and matching of pictorial structures. *IEEE Transactions on Computers.*, 1973.

[80] Daniel Fried, Jacob Andreas, and Dan Klein. Unified pragmatic models for generating and following instructions. *Computing Research Repository (CoRR)*, abs/1711.04987, 2017.

[81] Tsu-Jui Fu, Xin Eric Wang, and W. Wang. Language-driven image style transfer. *Computing Research Repository (CoRR)*, abs/2106.00178, 2021.

[82] Edward Gibson, Richard Futrell, Julian Jara-Ettinger, Kyle Mahowald, Leon Bergen, Sivalogeswaran Ratnasingam, Mitchell Gibson, Steven T. Piantadosi, and Bevil R. Conway. Color naming across languages reflects color use. *Proceedings of the National Academy of Sciences (PNAS)*, 2017.

[83] Rohit Girdhar, David F. Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *European Conference on Computer Vision (ECCV)*, 2016.

[84] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

[85] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[86] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, 2016.

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

[88] Noah D. Goodman and Michael C. Frank. Pragmatic language interpretation as probabilistic inference. *Trends in Cognitive Sciences*, 2016.

[89] Jonathan Gordon and José Miguel Hernández-Lobato. Combining deep generative and discriminative models for bayesian semi-supervised learning. *Pattern Recognition*, 2020.

[90] Ankit Goyal, Kaiyu Yang, Dawei Yang, and Jia Deng. Rel3D: A minimally contrastive benchmark for grounding spatial relations in 3D. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[91] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[92] Caroline Graf, Judith Degen, Robert X. D. Hawkins, and Noah D. Goodman. Animal, dog, or dalmatian? level of abstraction in nominal referring expressions. In *Proceedings of the 38th Annual Conference of the Cognitive Science Society*, 2016.

[93] H. P. Grice. Logic and conversation. In P. Cole and J. Morgan, editors, *Syntax and Semantics*, pages 43–58. Academic Press, New York, 1975.

[94] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. *Computing Research Repository (CoRR)*, abs/1704.00028, 2017.

[95] Songfang Han, Jiayuan Gu, Kaichun Mo, Li Yi, Siyu Hu, Xuejin Chen, and Hao Su. Compositionally generalizable 3d structure prediction. *Computing Research Repository (CoRR)*, abs/2012.02493, 2020.

[96] Ankur Handa, Viorica Patraucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. SceneNet: Understanding real world indoor scenes with synthetic data. *Computing Research Repository (CoRR)*, abs/1511.07041, 2015.

[97] Nils Hasler, Carsten Stoll, Martin Sunkel, Bodo Rosenhahn, and H-P Seidel. A statistical model of human pose and body shape. In *Computer graphics forum*, 2009.

[98] Robert X. D. Hawkins. Conducting real-time multiplayer experiments on the web. *Behavior Research Methods*, 2015.

[99] Robert X. D. Hawkins, Minae Kwon, Dorsa Sadigh, and Noah D. Goodman. Continual adaptation for efficient machine communication. *Computing Research Repository (CoRR)*, abs/1911.09896, 2019.

[100] Dailan He, Yusheng Zhao, Junyu Luo, Tianrui Hui, Shaofei Huang, Aixi Zhang, and Si Liu. TransRefer3D: Entity-and-relation aware transformer for fine-grained 3D visual grounding. *Computing Research Repository (CoRR)*, abs/2108.02388, 2021.

[101] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Computing Research Repository (CoRR)*, abs/1512.03385, 2015.

[102] Vishakh Hegde and Reza Zadeh. Fusionnet: 3d object classification using multiple data representations. *Computing Research Repository (CoRR)*, abs/1607.05695, 2016.

[103] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *Computing Research Repository (CoRR)*, abs/1506.05163, 2015.

[104] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations (ICLR)*, 2016.

[105] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 2006.

[106] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[107] Ji Hou, Benjamin Graham, Matthias Nießner, and Saining Xie. Exploring data-efficient 3d scene understanding with contrastive scene contexts. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[108] Ruizhen Hu, Zihao Yan, Jingwen Zhang, Oliver van Kaick, Ariel Shamir, Hao Zhang, and Hui Huang. Predictive and generative neural networks for object functionality. In *Computer Graphics Forum*, 2018.

[109] Ruqi Huang, **Panos Achlioptas**, Leonidas Guibas, and Maks Ovsjanikov. Latent space representation for shape analysis and learning. *Computing Research Repository (CoRR)*, 1806.03967, 2018.

[110] Ruqi Huang, **Panos Achlioptas**, Leonidas Guibas, and Maks Ovsjanikov. Limit Shapes – a tool for understanding shape differences and variability in 3d model collections. In *Eurographics Symposium on Geometry Processing (SGP)*, 2019.

[111] Ruqi Huang, Marie-Julie Rakotosaona, **Panos Achlioptas**, Leonidas J. Guibas, and Maks Ovsjanikov. OperatorNet: Recovering 3D shapes from difference operators. In *International Conference on Computer Vision (ICCV)*, 2019.

[112] Sample Ian. Computer says no: why making ais fair, accountable and transparent is crucial, 2017. The Guardian.

[113] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015.

[114] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. Deep unordered composition rivals syntactic methods for text classification. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2015.

[115] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[116] Wenzel Jakob. Mitsuba renderer, 2010. http://www.mitsuba-renderer.org.

[117] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. Fasttext.zip: Compressing text classification models. *Computing Research Repository (CoRR)*, abs/1612.03651, 2016.

[118] Evangelos Kalogerakis, Melinos Averkiou, Subhransu Maji, and Siddhartha Chaudhuri. 3d shape segmentation with projective convolutional networks. *Computing Research Repository (CoRR)*, abs/1612.02808, 2016.

[119] Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, and Vladlen Koltun. A probabilistic model for component-based shape synthesis. *ACM Transactions on Graphics (TOG)*, 2012.

[120] Angjoo Kanazawa, Shahar Kovalsky, Ronen Basri, and David Jacobs. Learning 3D deformation of animals from 2D images. In *Computer Graphics Forum*, 2016.

[121] Hamid Karimi, Tyler Derr, and Jiliang Tang. Characterizing the decision boundary of deep neural networks. *Computing Research Repository (CoRR)*, abs/1912.11460, 2020.

[122] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137, 2015.

[123] Andrej Karpathy, Armand Joulin, and Fei Fei Li. Deep fragment embeddings for bidirectional image sentence mapping. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[124] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *Computing Research Repository (CoRR)*, abs/1812.04948, 2018.

[125] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. *Computing Research Repository (CoRR)*, abs/1912.04958, 2019.

[126] Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. Referitgame: Referring to objects in photographs of natural scenes. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[127] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Rotation invariant spherical harmonic representation of 3d shape descriptors. In *Eurographics Symposium on Geometry Processing (SGP)*, 2003.

[128] Jin-Hwa Kim, Jaehyun Jun, and Byoung-Tak Zhang. Bilinear attention networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[129] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Computing Research Repository (CoRR)*, abs/1412.6980, 2014.

[130] Diederik P Kingma, Tim Salimans, and Max Welling. Improving variational inference with inverse autoregressive flow. *Computing Research Repository (CoRR)*, abs/1606.04934, 2016.

[131] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *Computing Research Repository (CoRR)*, abs/1312.6114, 2013.

[132] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *Computing Research Repository (CoRR)*, abs/1609.02907, 2016.

[133] Simon Kirby, Monica Tamariz, Hannah Cornish, and Kenny Smith. Compression and communication in the cultural evolution of linguistic structure. *Cognition*, 2015.

[134] Ryan Kiros, Ruslan Salakhutdinov, Richard S Zemel, and et al. Unifying visual-semantic embeddings with multimodal neural language models. *Transactions of the Association for Computational Linguistics (TACL)*, 2015.

[135] Chen Kong, Dahua Lin, Mohit Bansal, Raquel Urtasun, and Sanja Fidler. What are you talking about? text-to-image coreference. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[136] Artiom Kovnatsky, Michael M Bronstein, Alexander M Bronstein, Klaus Glashoff, and Ron Kimmel. Coupled quasi-harmonic bases. In *Computer Graphics Forum*, 2013.

[137] Robert M. Krauss and Sidney Weinheimer. Changes in reference phrases as a function of frequency of usage in social interaction: A preliminary study. *Psychonomic Science*, 1964.

[138] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome:

Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision (IJCV)*, 2017.

[139] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, https://www.cs.toronto.edu, 2009.

[140] Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. *Computing Research Repository (CoRR)*, abs/2010.07954, 2020.

[141] Volodymyr Kuleshov and S. Ermon. Deep hybrid models: Bridging discriminative and generative approaches. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2017.

[142] Nilesh Kulkarni, Ishan Misra, Shubham Tulsiani, and Abhinav Gupta. 3D-relnet: Joint object and relational network for 3D prediction. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[143] Solomon. Kullback and Richard A Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 1951.

[144] Andrey Kurenkov, Jingwei Ji, Animesh Garg, Viraj Mehta, JunYoung Gwak, Christopher Choy, and Silvio Savarese. Deformnet: Free-form deformation network for 3d shape reconstruction from a single image. In *Winter Conference on Applications of Computer Vision (WACV)*, 2018.

[145] Guillaume Lample, Neil Zeghidour, Nicolas Usunier, Antoine Bordes, Ludovic Denoyer, and Marc'Aurelio Ranzato. Fader Networks: Manipulating images by sliding attributes. *Computing Research Repository (CoRR)*, abs/1706.00409, 2018.

[146] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *Computing Research Repository (CoRR)*, abs/1512.09300, 2015.

[147] J.A. Lasserre, C.M. Bishop, and T.P. Minka. Principled hybrids of generative and discriminative models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[148] Angeliki Lazaridou, Karl Moritz Hermann, Karl Tuyls, and Stephen Clark. Emergence of linguistic communication from referential games with symbolic and pixel input. *Computing Research Repository (CoRR)*, abs/1804.03984, 2018.

[149] Truc Le and Ye Duan. PointGrid: A deep network for 3d shape understanding. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[150] Jimmy Lei Ba, Kevin Swersky, Sanja Fidler, et al. Predicting deep zero-shot convolutional neural networks using textual descriptions. In *International Conference on Computer Vision (ICCV)*, 2015.

[151] B. Levy. Laplace-beltrami eigenfunctions towards an algorithm that "understands" geometry. In *IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)*, pages 13–13, June 2006.

[152] Thomas Lewiner, Helio Lopes, Antonio Wilson Vieira, and Geovan Tavares. Efficient implementation of marching cubes' cases with topological guarantees. *Journal of Graphics Tools*, 2003.

[153] David Lewis. *Convention: A philosophical study*. Harvard University Press, 1969.

[154] Chengshu Li, Fei Xia, Roberto Martín-Martín, and Silvio Savarese. HRL4IN: Hierarchical reinforcement learning for interactive navigation with mobile manipulators. In *Conference on Robot Learning (CoRL)*, 2020.

[155] Jun Li, Chengjie Niu, and Kai Xu. Learning part generation and assembly for structure-aware shape synthesis. *Computing Research Repository (CoRR)*, abs/1906.06693, 2019.

[156] Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. Grass: Generative recursive autoencoders for shape structures. *ACM Transactions on Graphics (TOG)*, 2017.

[157] Ruihui Li, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. PU-GAN: A point cloud upsampling adversarial network. In *International Conference on Computer Vision (ICCV)*, 2019.

[158] Yujia Li, Kevin Swersky, and Rich Zemel. Generative moment matching networks. In *International Conference on Machine Learning (ICML)*, 2015.

[159] Chen-Hsuan Lin, Ersin Yumer, Oliver Wang, Eli Shechtman, and Simon Lucey. St-gan: Spatial transformer generative adversarial networks for image compositing. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[160] Or Litany, Alex Bronstein, Michael Bronstein, and Ameesh Makadia. Deformable shape completion with graph convolutional autoencoders. In *Proc. CVPR*, pages 1886–1895, 2018.

[161] Or Litany, Tal Remez, Emanuele Rodolà, Alex Bronstein, and Michael Bronstein. Deep functional maps: Structured prediction for dense shape correspondence. In *Proc. ICCV*, pages 5659–5667, 2017.

[162] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *International Conference on Computer Vision (ICCV)*, 2015.

[163] Francesco Locatello, Stefan Bauer, Mario Lucic, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. *International Conference on Machine Learning (ICML)*, 2019.

[164] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[165] Yang Long and Ling Shao. Describing unseen classes by exemplars: Zero-shot learning using grouped simile ensemble. In *Winter Conference on Applications of Computer Vision (WACV)*, 2017.

[166] Yang Long and Ling Shao. Learning to recognise unseen classes by a few similes. In *ACM International Conference on Multimedia (MM)*, 2017.

[167] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Smpl: A skinned multi-person linear model. *ACM Trans. Graph.*, 34(6):248:1–248:16, October 2015.

[168] Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. Visual relationship detection with language priors. In *European Conference on Computer Vision (ECCV)*, 2016.

[169] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Neural baby talk. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[170] Ruotian Luo and Gregory Shakhnarovich. Comprehension-guided referring expressions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[171] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *Computing Research Repository (CoRR)*, abs/1508.04025, 2015.

[172] Rui Ma, Akshay Gadi Patil, Matthew Fisher, Manyi Li, Sören Pirk, Binh-Son Hua, Sai-Kit Yeung, Xin Tong, Leonidas Guibas, and Hao Zhang. Language-driven synthesis of 3D scenes from scene databases. *ACM SIGGRAPH Asia*, 2018.

[173] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *International Conference on Machine Learning (ICML)*, 2013.

[174] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research (JMLR)*, 2008.

[175] Maierdan Maimaitimin, Keigo Watanabe, and Shoichi Maeyama. Stacked convolutional auto-encoders for surface recognition based on 3d point cloud data. *Artificial Life and Robotics*, 2017.

[176] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *Computing Research Repository (CoRR)*, abs/1511.05644, 2015.

[177] Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan Yuille, and Murphy Kevin. Generation and comprehension of unambiguous object descriptions. *CoRR*, abs/1511.02283, 2016.

[178] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, and Alan Yuille. Deep captioning with multimodal recurrent neural networks (m-rnn). In *International Conference on Learning Representations (ICLR)*, 2015.

[179] Haggai Maron, Meirav Galun, Noam Aigerman, Miri Trope, Nadav Dym, Ersin Yumer, VLADIMIR G KIM, and Yaron Lipman. Convolutional neural networks on surfaces via seamless toric covers. In *ACM Transactions on Graphics (TOG)*, 2017.

[180] Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *Proc. ICCV workshops*, 2015.

[181] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 922–928. IEEE, 2015.

[182] Cecilia Mauceri, Martha Palmer, and Christoffer Heckman. SUN-Spot: An RGB-D dataset with spatial referring expressions. In *International Conference on Computer Vision Workshop on Closing the Loop Between Vision and Language*, 2019.

[183] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H Barr. Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. In *Visualization and mathematics III*, pages 35–57. Springer, 2003.

[184] David Mickisch, Felix Assion, Florens Greßner, Wiebke Günther, and Mariele Motta. Understanding the decision boundary of deep neural networks: An empirical study. *Computing Research Repository (CoRR)*, abs/2002.01810, 2020.

[185] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *Computing Research Repository (CoRR)*, abs/1301.3781, 2013.

[186] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *Computing Research Repository (CoRR)*, abs/1310.4546, 2013.

[187] Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. Deep learning based text classification: A comprehensive review. *Computing Research Repository (CoRR)*, abs/2004.03705, 2020.

[188] Margaret Mitchell, Kees van Deemter, and Ehud Reiter. Generating expressions that refer to visible objects. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, 2013.

[189] Takeru Miyato, Dai M. Andrew, and Goodfellow Ian. Adversarial training methods for semi-supervised text classification. *International Conference on Learning*, 2017.

[190] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent models of visual attention. *Computing Research Repository (CoRR)*, abs/1406.6247, 2014.

[191] Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy J. Mitra, and Leonidas J. Guibas. Structurenet: Hierarchical graph networks for 3d shape generation. *Computing Research Repository (CoRR)*, abs/1908.00575, 2019.

[192] Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[193] Will Monroe, Robert XD Hawkins, Noah D Goodman, and Christopher Potts. Colors in context: A pragmatic neural model for grounded language understanding. *Transactions of the Association for Computational Linguistics (TACL)*, 2017.

[194] K. Varun Nagaraja, I. Vlad Morariu, and Davis S. Larry. Modeling context between objects for referring expression understanding. *European Conference on Computer Vision (ECCV)*, 2016.

[195] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning (ICML)*, 2010.

[196] Charlie Nash and Chris KI Williams. The shape variational autoencoder: A deep generative model of part-segmented 3d objects. In *Computer Graphics Forum*, 2017.

[197] Andy Nguyen, Mirela Ben-Chen, Katarzyna Welnicka, Yinyu Ye, and Leonidas Guibas. An optimization approach to improving collections of shape maps. In *Computer Graphics Forum*, 2011.

[198] Anh Nguyen, Jason Yosinski, Yoshua Bengio, Alexey Dosovitskiy, and Jeff Clune. Plug & play generative networks: Conditional iterative generation of images in latent space. *Computing Research Repository (CoRR)*, abs/1612.00005, 2016.

[199] Fakir S. Nooruddin and Greg Turk. Simplification and repair of polygonal models using volumetric techniques. *IEEE Transactions on Visualization and Computer Graphics*, 2003.

[200] D. Nyga, Subhro Roy, Rohan Paul, Daehyung Park, M. Pomarlan, M. Beetz, and N. Roy. Grounding robot plans from natural language instructions with incomplete world knowledge. In *Conference on Robot Learning (CoRL)*, 2018.

[201] Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. Functional Maps: A flexible representation of maps between shapes. *ACM Transactions on Graphics (TOG)*, 2012.

[202] Maks Ovsjanikov, Etienne Corman, Michael Bronstein, Emanuele Rodolà, Mirela Ben-Chen, Leonidas Guibas, Frederic Chazal, and Alex Bronstein. Computing and processing correspondences with functional maps. In *ACM SIGGRAPH 2017 Courses*, 2017.

[203] Maike Paetzel, David Nicolas Racca, and David DeVault. A multimodal corpus of rapid dialogue games. In *International Conference on Language Resources and Evaluation (LREC)*, 2014.

[204] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A method for automatic evaluation of machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2002.

[205] D. Parikh and K. Grauman. Relative attributes. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[206] Razvan Pascanu, Tomás Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. In *International Conference on Machine Learning (ICML)*, 2012.

[207] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[208] Claudia Pérez-D'Arpino and Julie A. Shah. C-LEARN: Learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[209] Han-Hung Lee Pin-Hao Huang, Hwann-Tzong Chen, and Tyng-Luh Liu. Text-guided graph neural networks for referring 3D instance segmentation. In *AAAI Conference on Artificial Intelligence*, 2021.

[210] Ulrich Pinkall and Konrad Polthier. Computing Discrete Minimal Surfaces and their Conjugates. *Experimental mathematics*, 2(1):15–36, 1993.

[211] Bryan A Plummer, Kevin J Shih, Yichen Li, Ke Xu, Svetlana Lazebnik, Stan Sclaroff, and Kate Saenko. Revisiting image-language networks for open-ended phrase detection. *Computing Research Repository (CoRR)*, abs/1811.07212, 2018.

[212] Bryan A Plummer, Liwei Wang, Chris M Cervantes, Juan C Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for

richer image-to-sentence models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[213] Christian Poelitz. Projection based transfer learning. In *Workshops at ECML*, 2014.

[214] Dominic V Poerio and Steven D Brown. Dual-domain calibration transfer using orthogonal projection. *Applied spectroscopy*, 2018.

[215] Mihir Prabhudesai, Hsiao-Yu Fish Tung, Syed Ashar Javed, Maximilian Sieb, Adam W Harley, and Katerina Fragkiadaki. Embodied language grounding with implicit 3D visual feature representations. *Computing Research Repository (CoRR)*, abs/1910.01210, 2019.

[216] Albert Pumarola, Antonio Agudo, Aleix M Martinez, Alberto Sanfeliu, and Francesc Moreno-Noguer. Ganimation: Anatomically-aware facial animation from a single image. In *European Conference on Computer Vision (ECCV)*, 2018.

[217] C. Qi, Xinlei Chen, O. Litany, and L. Guibas. ImVoteNet: Boosting 3D object detection in point clouds with image votes. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[218] Charles R. Qi, , Or Litany, Kaiming He, and Leonidas Guibas. Deep hough voting for 3D object detection in point clouds. In *International Conference on Computer Vision (ICCV)*, 2019.

[219] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[220] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proc. CVPR*, pages 5648–5656, 2016.

[221] Charles Ruizhongtai Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[222] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[223] Fengchun Qiao, Naiming Yao, Zirui Jiao, Zhihao Li, Hui Chen, and Hongan Wang. Geometry-contrastive gan for facial expression transfer. *Computing Research Repository (CoRR)*, abs/1802.01822, 2018.

[224] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *Computing Research Repository (CoRR)*, abs/2103.00020, 2021.

[225] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *Computing Research Repository (CoRR)*, abs/1511.06434, 2015.

[226] V. Ramanathan, A. Joulin, P. Liang, and L. Fei-Fei. Linking people with "their" names using coreference resolution. In *European Conference on Computer Vision (ECCV)*, 2014.

[227] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *Computing Research Repository (CoRR)*, abs/2102.12092, 2021.

[228] Mengye Ren, Ryan Kiros, and Richard Zemel. Exploring models and data for image question answering. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[229] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[230] Salah Rifai, Yoshua Bengio, Yann Dauphin, and Pascal Vincent. A generative process for sampling contractive auto-encoders. *Computing Research Repository (CoRR)*, abs/1206.6434, 2012.

[231] Junha Roh, Karthik Desingh, Ali Farhadi, and Dieter Fox. LanguageRefer: Spatial-language model for 3D visual grounding. *Computing Research Repository (CoRR)*, abs/2107.03438, 2021.

[232] Bernardino Romera-Paredes and Philip Torr. An embarrassingly simple approach to zero-shot learning. In *International Conference on Machine Learning (ICML)*, 2015.

[233] Seymour Rosenberg and Bertram D. Cohen. Speakers' and listeners' processes in a word-communication task. *Science*, 1964.

[234] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 1958.

[235] Benjamin Rosman and Subramanian Ramamoorthy. Learning spatial relationships between objects. *The International Journal of Robotics Research*, 2011.

[236] Jean-Michel Roufosse and Maks Ovsjanikov. Unsupervised deep learning for structured shape matching. *CoRR*, abs/1812.03794, 2018.

[237] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision (IJCV)*, 2000.

[238] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 1988.

[239] Raif M. Rustamov, Maks Ovsjanikov, Omri Azencot, Mirela Ben-Chen, Frédéric Chazal, and Leonidas Guibas. Map-based exploration of intrinsic shape differences and variability. *ACM Transactions on Graphics (TOG)*, 2013.

[240] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[241] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[242] Nadav Schor, Oren Katzir, Hao Zhang, and Daniel Cohen-Or. CompoNet: Learning to generate the unseen by part synthesis and composition. *International Conference on Computer Vision (ICCV)*, 2019.

[243] Adriana Schulz, Ariel Shamir, Ilya Baran, David I. W. Levin, Pitchaya Sitthi-Amorn, and Wojciech Matusik. Retrieval on parametric shape collections. *ACM Trans. Graph.*, 36(4), January 2017.

[244] Lutfi Kerem Senel, Ihsan Utlu, Veysel Yucesoy, Aykut Koc, and Tolga Cukur. Semantic structure and interpretability of word embeddings. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2018.

[245] Abhishek Sharma, O. Grau, and Mario Fritz. Vconv-dae: Deep volumetric shape learning without object labels. *Computing Research Repository (CoRR)*, abs/1604.03755, 2016.

[246] Chao-Hui Shen, Hongbo Fu, Kang Chen, and Shi-Min Hu. Structure recovery by part assembly. *ACM Transactions on Graphics (TOG)*, 2012.

[247] Sheng Shen and Hung Lee. Neural attention models for sequence classification: Analysis and application to key term extraction and dialogue act detection. *Computing Research Repository (CoRR)*, abs/1604.00077, 2016.

[248] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. *Computing Research Repository (CoRR)*, abs/1912.01734, 2019.

[249] Zhixin Shu, Mihir Sahasrabudhe, Alp Guler, Dimitris Samaras, Nikos Paragios, and Iasonas Kokkinos. Deforming autoencoders: Unsupervised disentangling of shape and appearance. *Computing Research Repository (CoRR)*, 2018.

[250] Zhixin Shu, Ersin Yumer, Sunil Hadap, Kalyan Sunkavalli, Eli Shechtman, and Dimitris Samaras. Neural face editing with intrinsic image disentangling. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[251] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *Computing Research Repository (CoRR)*, abs/1409.1556, 2014.

[252] A. Sinha, A. Unmesh, Q. Huang, and K. Ramani. Surfnet: Generating 3d shape surfaces using deep residual networks. In *CVPR*, pages 791–800, July 2017.

[253] Ayan Sinha, Jing Bai, and Karthik Ramani. Deep learning 3d shape surfaces using geometry images. In *European Conference on Computer Vision*, pages 223–240. Springer, 2016.

[254] Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. Zero-shot learning through cross-modal transfer. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

[255] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2013.

[256] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. How to train deep variational autoencoders and probabilistic ladder networks. *Computing Research Repository (CoRR)*, abs/1602.02282, 2016.

[257] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 2014.

[258] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *International Conference on Computer Vision (ICCV)*, 2015.

[259] Jong-Chyi Su, Chenyun Wu, Huaizu Jiang, and Subhransu Maji. Reasoning about fine-grained attribute phrases using reference games. In *International Conference on Computer Vision (ICCV)*, 2017.

[260] Yongbin Sun, Yue Wang, Ziwei Liu, Joshua Siegel, and Sanjay Sarma. PointGrow: Autoregressively learned point cloud generation with self-attention. In *Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[261] Minhyuk Sung, Zhenyu Jiang, **Panos Achlioptas**, N. Mitra, and L. Guibas. DeformSyncNet: Deformation transfer via synchronized shape deformation spaces. *ACM Transactions on Graphics (TOG)*, 2020.

[262] Minhyuk Sung, Vladimir G Kim, Roland Angst, and Leonidas J. Guibas. Data-driven structural priors for shape completion. *ACM Transactions on Graphics (TOG)*, 2015.

[263] Minhyuk Sung, Hao Su, Vladimir Kim, Siddhartha Chaudhuri, and and Guibas. ComplementMe: Weakly-supervised component suggestions for 3D modeling. In *ACM Transactions on Graphics (TOG)*, 2017.

[264] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *Computing Research Repository (CoRR)*, abs/1512.00567, 2015.

[265] Flora Ponjou Tasse and Neil Dodgson. Shape2Vec: Semantic-based descriptors for 3d shapes, sketches and images. *ACM Transactions on Graphics (TOG)*, 2016.

[266] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. *Computing Research Repository (CoRR)*, abs/1703.09438, 2017.

[267] Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew Walter, Ashis Banerjee, Seth Teller, and Nicholas Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI Conference on Artificial Intelligence*, 2011.

[268] J. Tenenbaum, Charles Kemp, T. Griffiths, and Noah D. Goodman. How to grow a mind: Statistics, structure, and abstraction. *Science*, 2011.

[269] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *Computing Research Repository (CoRR)*, abs/1906.05849, 2019.

[270] Yao-Hung Hubert Tsai, Liang-Kang Huang, and Ruslan Salakhutdinov. Learning robust visual-semantic embeddings. In *International Conference on Computer Vision (ICCV)*, 2017.

[271] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. *Computing Research Repository (CoRR)*, abs/1908.04616, 2019.

[272] Kees van Deemter. *Computational models of referring: a study in cognitive science*. MIT Press, 2016.

[273] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *Computing Research Repository (CoRR)*, abs/1807.03748, 2018.

[274] Gül Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. In *CVPR*, 2017.

[275] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[276] Ramakrishna Vedantam, Samy Bengio, Kevin Murphy, Devi Parikh, and Gal Chechik. Context-aware captions from context-agnostic supervision. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[277] Evangelos Ververas and Stefanos Zafeiriou. Slidergan: Synthesizing expressive face images by sliding 3d blendshape parameters. In *International Journal of Computer Vision (IJCV)*, 2020.

[278] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[279] Fan Wang, Qixing Huang, and Leonidas J Guibas. Image co-segmentation via consistent functional maps. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[280] Hao Wang, Nadav Schor, Ruizhen Hu, Haibin Huang, Daniel Cohen-Or, and Hui Huang. Global-to-local generative model for 3d shapes. *ACM SIGGRAPH Asia*, 2018.

[281] He Wang, Zetian Jiang, Li Yi, Kaichun Mo, Hao Su, and Leonidas J. Guibas. Rethinking sampling in 3D point cloud generative adversarial networks. *Computing Research Repository (CoRR)*, abs/2006.07029, 2020.

[282] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions on Graphics (TOG)*, 2017.

[283] Sida I Wang and Christopher D Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2012.

[284] Xiaolong Wang and Abhinav Gupta. Generative image modeling using style and structure adversarial networks. In *European Conference on Computer Vision (ECCV)*, 2016.

[285] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 2019.

[286] Yueqing Wang, Zhige Xie, Kai Xu, Yong Dou, and Yuanwu Lei. An efficient and effective convolutional auto-encoder extreme learning machine network for 3d feature learning. *Neurocomputing*, 2016.

[287] Lingyu Wei, Qixing Huang, Duygu Ceylan, Etienne Vouga, and Hao Li. Dense human body correspondences using convolutional networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[288] Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1989.

[289] Ludwig Wittgenstein. *Philosophical Investigations: The English Text of the Third Edition*. Macmillan, 1953.

[290] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[291] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *Computing Research Repository (CoRR)*, abs/1609.08144, 2016.

[292] Zhijie Wu, Xiang Wang, Di Lin, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. SAGNet: Structure-aware generative network for 3d-shape modeling. *ACM SIGGRAPH*, 2019.

[293] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[294] Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance-level discrimination. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[295] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[296] Yongqin Xian, Christoph H. Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning – a comprehensive evaluation of the good, the bad and the ugly. *Computing Research Repository (CoRR)*, abs/1707.00600, 2020.

[297] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel Chang, Leonidas Guibas, and Hao Su. SAPIEN: A simulated part-based interactive environment. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[298] Tianjun Xiao, Yichong Xu, Kuiyuan Yang, Jiaxing Zhang, Yuxin Peng, and Zheng Zhang. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[299] Saining Xie, Jiatao Gu, Demi Guo, Charles R. Qi, Leonidas J. Guibas, and Or Litany. PointContrast: Unsupervised pre-training for 3D point cloud understanding. *Computing Research Repository (CoRR)*, abs/2007.10985, 2020.

[300] Kai Xu, Hanlin Zheng, Hao Zhang, Daniel Cohen-Or, Ligang Liu, and Yueshan Xiong. Photo-inspired model-driven 3d object modeling. In *ACM Transactions on Graphics (TOG)*, 2011.

[301] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. *Computing Research Repository (CoRR)*, abs/1502.03044, 2016.

[302] Kelvin Xu, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning (ICML)*, 2015.

[303] Jianwei Yang, Zhile Ren, Mingze Xu, Xinlei Chen, David Crandall, Devi Parikh, and Dhruv Batra. Embodied visual recognition. *Computing Research Repository (CoRR)*, abs/1904.04404, 2019.

[304] Yongxin Yang and Timothy M Hospedales. A unified perspective on multi-domain and multi-task learning. In *International Conference on Learning Representations (ICLR)*, 2015.

[305] Ze Yang, Tiange Luo, Dong Wang, Zhiqiang Hu, Jun Gao, and Liwei Wang. Learning to navigate for fine-grained classification. *Computing Research Repository (CoRR)*, abs/1809.00287, 2018.

[306] Zhengyuan Yang, Songyang Zhang, Liwei Wang, and Jiebo Luo. SAT: 2d semantics assisted training for 3D visual grounding. *Computing Research Repository (CoRR)*, abs/2105.11450, 2021.

[307] Li Yi, Vladimir G. Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas J. Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (TOG)*, 2016.

[308] Li Yi, Hao Su, Xingwen Guo, and Leonidas J. Guibas. SyncSpecCNN: Synchronized spectral CNN for 3d shape segmentation. *Computing Research Repository (CoRR)*, abs/1612.00606, 2016.

[309] A. Yu and K. Grauman. Fine-grained visual comparisons with local learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[310] A. Yu and K. Grauman. Semantic jitter: Dense supervision for visual comparisons via synthetic images. In *International Conference on Computer Vision (ICCV)*, 2017.

[311] Licheng Yu, Zhe Lin, Xiaohui Shen, Yangm Jimei, Xin Lu, Mohit Bansal, and L. Tamara Berg. Mattnet: Modular attention network for referring expression comprehension. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[312] Licheng Yu, Patrick Poirson, Shan Yang, C. Alexander Berg, and L. Tamara Berg. Modeling context in referring expressions. *European Conference on Computer Vision (ECCV)*, 2016.

[313] Licheng Yu, Hao Tan, Mohit Bansal, and Tamara L. Berg. A joint speaker-listener-reinforcer model for referring expressions. *Computing Research Repository (CoRR)*, abs/1612.09542, 2017.

[314] Yi Yu, Abhishek Srivastava, and Simon Canales. Conditional lstm-gan for melody generation from lyrics. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 2021.

[315] Zhou Yu, Jun Yu, Yuhao Cui, Dacheng Tao, and Qi Tian. Deep modular co-attention networks for visual question answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[316] Zhihao Yuan, Xu Yan, Yinghong Liao, Ruimao Zhang, Zhen Li, and Shuguang Cui. InstanceRefer: Cooperative holistic understanding for visual grounding on point clouds through instance multi-level contextual referring. *International Conference on Computer Vision (ICCV)*, 2021.

[317] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola. Deep sets. *Computing Research Repository (CoRR)*, 2017.

[318] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *Computing Research Repository (CoRR)*, abs/1805.08318, 2019.

[319] Ning Zhang, Jeff Donahue, Ross Girshick, and Trevor Darrell. Part-based r-cnns for fine-grained category detection. In *European Conference on Computer Vision (ECCV)*, 2014.

[320] Han Zhao, Zhenyao Zhu, Junjie Hu, Adam Coates, and Geoff Gordon. Principled hybrids of generative and discriminative domain adaptation. *Computing Research Repository (CoRR)*, abs/1705.09011, 2017.

[321] Yongheng Zhao, Tolga Birdal, Haowen Deng, and Federico Tombari. 3D point-capsule networks. *Computing Research Repository (CoRR)*, abs/1812.10775, 2018.

[322] Luowei Zhou, Yannis Kalantidis, Xinlei Chen, Jason J Corso, and Marcus Rohrbach. Grounded video description. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[323] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *International Conference on Computer Vision (ICCV)*, 2017.

[324] Long Zhu, Yuanhao Chen, Alan Yuille, and William Freeman. Latent hierarchical structural learning for object detection. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[325] Zhuotun Zhu, Xinggang Wang, Song Bai, Cong Yao, and Xiang Bai. Deep learning representation using autoencoder for 3d shape retrieval. *Neurocomputing*, 2016.

[326] Silvia Zuffi, Angjoo Kanazawa, David Jacobs, and Michael J. Black. 3D menagerie: Modeling the 3D shape and pose of animals. In *CVPR)*, July 2017.

Appendix A

AutoEncoders and GANs for 3D Point Clouds Objects

A.1 AE Details

The encoding layers of our AEs were implemented as 1D-convolutions with ReLUs, with kernel size of 1 and stride of 1, i.e. treating each 3D point independently. Their decoding layers, were MLPs built with FC-ReLUs. We used Adam [129] with initial learning rate of 0.0005, β_1 of 0.9 and a batch size of 50 to train all AEs.

A.1.1 AE used for SVM-based experiments

For the AE mentioned in the SVM-related experiments of Section 2.7.1 of the thesis, we used an encoder with $128, 128, 256$ and 512 filters in each of its layers and a decoder with $1024, 2048, 2048 \times 3$ neurons, respectively. Batch normalization was used between every layer. We also used online data augmentation by applying random rotations along the gravity-(z)-axis to the input point clouds of each batch. We trained this AE for 1000 epochs with the CD loss and for 1100 with the EMD.

A.1.2 All other AEs

For all other AEs, the encoder had $64, 128, 128, 256$ and k filters at each layer, with k being the bottle-neck size. The decoder was comprised by 3 FC-ReLU layers with $256, 256, 2048 \times 3$ neurons each. We trained these AEs for a maximum of 500 epochs when using single class data and 1000 epochs for the experiment involving 5 shape classes (end of Section 2.7.4, thesis).

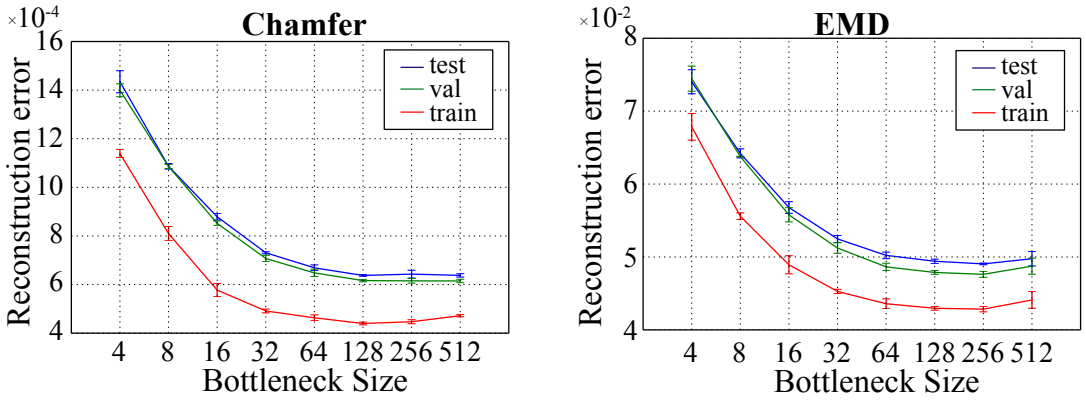


Figure A.1: The bottleneck size was fixed at 128 in all single-class experiments by observing the reconstruction loss of the AEs, shown here for various bottleneck sizes, when training with the data of the chair class.

A.1.3 AE regularization

To determine an appropriate size for the latent-space, we constructed 8 (otherwise architecturally identical) AEs with bottleneck sizes $k \in \{4, 8 \dots, 512\}$ and trained them with point clouds of the chair object class, under the two losses (Fig. A.1). We repeated this procedure with pseudo-random weight initializations three times and found that $k = 128$ had the best generalization error on the test data, while achieving minimal reconstruction error on the train split.

Remark. Different AE setups brought no noticeable advantage over our main architecture. Concretely, adding drop-out layers resulted in worse reconstructions and using batch-norm on the encoder *only*, sped up training and gave us slightly better generalization error when the AE was trained with single-class data. Exclusively, for the SVM experiment of Section 5.1 of the thesis we randomly rotate the input chairs to promote latent features that are rotation-invariant.

A.2 SVM Parameters for Auto-encoder Evaluation

For the classification experiments of Section 5.1 (thesis) we used a one-versus-rest linear SVM classifier with an l_2 norm penalty and balanced class weights. The exact optimization parameters can be found in Table A.1. The confusion matrix of the classifier evaluated on our latent codes on ModelNet40 is shown in Fig. A.2.

Loss	ModelNet40			ModelNet10		
	C-plt	icpt	loss	C-plt	icpt	loss
EMD	0.09	0.5	hng	0.02	3	sq-hng
CD	0.25	0.4	sq-hng	0.05	0.2	sq-hng

Table A.1: Training parameters of SVMs used in each dataset with each structural loss of the AE. *C-penalty (C-plt)*: term controlling the trade-off between the size of the learned margin and the misclassification rate; *intercept (icpt)*: extra dimension appended on the input features to center them; *loss*: svm's optimization loss function: hinge (*hng*), or squared-hinge (*sq-hng*).

A.3 r-GAN Details

The discriminator's first 5 layers are 1D-convolutions with stride/kernel of size 1 and $\{64, 128, 256, 256, 512\}$ filters each; interleaved with leaky-ReLU. They are followed by a feature-wise max-pool. The last 2 FC-leaky-ReLU layers have $\{128, 64\}$, neurons each and they lead to single sigmoid neuron. We used 0.2 units of leak.

The generator consists of 5 FC-ReLU layers with $\{64, 128, 512, 1024, 2048 \times 3\}$ neurons each. We trained r-GAN with Adam with an initial learning rate of 0.0001, and $beta_1$ of 0.5 in batches of size 50. The noise vector was drawn by a spherical Gaussian of 128 dimensions with zero mean and 0.2 units of standard deviation.

Some synthetic results produced by the r-GAN are shown in Fig. 2.10.

A.4 l-GAN Details

The discriminator consists of 2 FC-ReLU layers with $\{256, 512\}$ neurons each and a final FC layer with a single sigmoid neuron. The generator consists of 2 FC-ReLUs with $\{128, k = 128\}$ neurons each. When used the l-Wasserstein-GAN, we used a gradient penalty regularizer $\lambda = 10$ and trained the critic for 5 iterations for each training iteration of the generator. The training parameters (learning rate, batch size) and the generator's noise distribution were the same as those used for the r-GAN.

A.4.1 Model selection of GANs

All GANs are trained for maximally 2000 epochs; for each GAN, we select one of its training epochs to obtain the "final" model, based on how well the synthetic results match the ground-truth distribution. Specifically, at a given epoch, we use the GAN to generate a set of synthetic point clouds, and measure the distance between this set and the validation set. We avoid measuring this

distance using MMD-EMD, given the high computational cost of EMD. Instead, we use either the JSD or MMD-CD metrics to compare the synthetic dataset to the validation dataset. To further reduce the computational cost of model selection, we only check every 100 epochs (50 for r-GAN). The generalization error of the various GAN models, at various training epochs, as measured by MMD and JSD is shown in Fig. A.3 (left and middle).

Using the same JSD criterion, we also select the number and covariance type of Gaussian components for the GMM (Fig. A.4, left), and obtain the optimal value of 32 components. GMMs performed much better with full (as opposed to diagonal) covariance matrices, suggesting strong correlations between the latent dimensions (Fig. A.4, right).

When using MMD-CD as the selection criterion, we obtain models of similar quality and at similar stopping epochs (see Table A.2); the optimal number of Gaussians in this case was 40. The training behavior measured using MMD-CD can be seen in Fig. A.3 (right).

Method	Epoch	JSD	MMD-CD	MMD-EMD	COV-EMD	COV-CD
A	1350	0.1893	0.0020	0.1265	19.4	54.7
B	300	0.0463	0.0020	0.0800	32.6	58.2
C	200	0.0319	0.0022	0.0684	57.6	58.7
D	1700	0.0240	0.0020	0.0664	64.2	64.7
E	-	**0.0182**	**0.0018**	**0.0646**	**68.6**	**69.3**

Table A.2: Evaluation of five generators on **test-split** of *chair* data on epochs/models that were selected via minimal MMD-CD on the validation-split. We report: A: r-GAN, B: l-GAN (AE-CD), C: l-GAN (AE-EMD) , D: l-WGAN (AE-EMD), E: GMM-40-F (AE-EMD). GMM-40-F stands for a GMM with 40 Gaussian components with full covariances. The reported scores are averages of three pseudo-random repetitions. Compare this with Table 2.4. Note that the overall quality of the selected models remains the same, irrespective of the metric used for model selection.

A.5 Voxel AE Details

Our voxel-based AEs are fully-convolutional with the encoders consisting of 3D-Conv-ReLU layers and the decoders of 3D-Conv-ReLU-transpose layers. Below, we list the parameters of consecutive layers, listed left-to-right. The layer parameters are denoted in the following manner: (number of filters, filter size). Each Conv/Conv-Transpose has a stride of 2 except the last layer of the 32^3 decoder which has 4. In the last layer of the decoders we do not use a non-linearity. The abbreviation "bn" stands for batch-normalization.

- 32^3 - model

  Encoder: Input $\rightarrow$ (32, 6) $\rightarrow$ (32, 6) $\rightarrow$ bn $\rightarrow$ (64, 4) $\rightarrow$ (64, 2) $\rightarrow$ bn $\rightarrow$ (64, 2)

  Decoder: (64, 2) $\rightarrow$ (32, 4) $\rightarrow$ bn $\rightarrow$ (32, 6) $\rightarrow$ (1, 8) $\rightarrow$ Output

- 64^3 - model

  Encoder: Input $\rightarrow$ (32, 6) $\rightarrow$ (32, 6) $\rightarrow$ bn $\rightarrow$ (64, 4) $\rightarrow$ (64, 4) $\rightarrow$ bn $\rightarrow$ (64, 2) $\rightarrow$ (64, 2)

  Decoder: (64, 2) $\rightarrow$ (32, 4) $\rightarrow$ bn $\rightarrow$ (32, 6) $\rightarrow$ (32, 6) $\rightarrow$ bn $\rightarrow$ (32, 8) $\rightarrow$ (1, 8) $\rightarrow$ Output

We train each AE for 100 epochs with Adam under the binary cross-entropy loss. The learning rate was 0.001, the β_1 0.9 and the batch size 64. To validate our voxel AE architectures, we compared them in terms of reconstruction quality to the state-of-the-art method of [266] and obtained comparable results, as demonstrated in Table A.3.

Voxel Resolution	32	64
Ours	92.7	88.4
[266]	93.9	90.4

Table A.3: Reconstruction quality statistics for our dense voxel-based AE and the one of [266] for the ShapeNetCars dataset. Both approaches use a 0.5 occupancy threshold and the train-test split of [266]. Reconstruction quality is measured by measuring the intersection-over-union between the input and synthesized voxel grids, namely the ratio between the volume in the voxel grid that is 1 in both grids divided by the volume that is 1 in at least one grid.

Sample Set Size	COV-CD	MMD-CD	COV-EMD	MMD-EMD
Entire —Train—	97.3	0.0013	98.2	0.0545
1 × —Test—	54.0	0.0023	51.9	0.0699
3 × —Test—	79.4	0.0018	78.6	0.0633
Full-GMM/32 (3 × —Test—)	68.9	0.0018	67.4	0.0651

Table A.4: Quantitative results of a baseline sampling/memorizing model, for different sizes of sets sampled from the training set and evaluated against the test split. The first three rows show results of a memorizing model, while the third row corresponds to our generative model. The first row shows the results of memorizing the entire training chair dataset. The second and third rows show the averages of three repetitions of the sub-sampling procedure with different random seeds.

A.5.1 Memorization baseline

Here we compare our GMM-generator against a model that memorizes the training data of the chair class. To do this, we either consider the entire training set or randomly sub-sample it, to create sets of different sizes. We then evaluate our metrics between these "memorized" sets and the point clouds of test split (see Table A.4). The coverage/fidelity obtained by our generative models (last row) is slightly lower than the equivalent in size case (third row) as expected: memorizing the training set produces good coverage/fidelity with respect to the test set when they are both drawn from the same population. This speaks for the validity of our metrics. Naturally, the advantage of using a learned representation lies in learning the structure of the underlying space instead of individual samples, which enables compactly representing the data and generating novel shapes as demonstrated by our interpolations. In particular, note that while some mode collapse is present in our generative results, as indicated by the $\sim 10\%$ drop in coverage, the MMD of our generative models is almost identical to that of the memorization case, indicating excellent fidelity.

A.6 More Comparisons with Wu et al.

In addition to the EMD-based comparisons in Table 4 of the thesis, in Tables A.5, A.6 we provide comparisons with [293] for the ShapeNet classes for which the authors have made publicly available their models. In Table A.5 we provide JSD-based comparisons for two of our models. In Table A.6 we provide Chamfer-based Fidelity/Coverage comparisons on the *test* split, that complement the EMD-based ones of Table 4 in the thesis.

Class	A	B		C	
	Tr+Te	Tr	Te	Tr	Te
airplane	-	0.0149	0.0268	**0.0065**	0.0191
car	0.1890	0.0081	0.0109	**0.0063**	0.0108
rifle	0.2012	0.0212	0.0364	**0.0092**	0.0214
sofa	0.1812	**0.0102**	0.0102	**0.0102**	0.0101
table	0.2472	0.0058	0.0177	**0.0035**	0.0143

Table A.5: JSD-based comparison between. TODO

Comparisons on training data. In Table A.7 we compare to [290] in terms of the JSD and MMD-CD on the training set of the *chair* category. Since [290] do not use any train/test split, we perform 5

Class	MMD-CD		COV-CD	
	A	B	A	B
airplane	-	0.0005	-	71.1
car	0.0015	**0.0007**	22.9	**63.0**
rifle	0.0008	**0.0005**	56.7	**71.7**
sofa	0.0027	**0.0013**	42.4	**75.5**
table	0.0058	**0.0016**	16.7	**71.7**

Table A.6: Chamfer-based MMD and Coverage comparison among methods

rounds of sampling 1K synthetic results from their models and report the best values of the respective evaluation metrics. We also report the average classification probability of the synthetic samples to be classified as chairs by the PointNet classifier. The r-GAN mildly outperforms [290] in terms of its diversity (as measured by JSD/MMD), while also creating realistic-looking results, as shown by the classification score. The l-GANs perform even better, both in terms of classification and diversity, with less training epochs. Finally, note that the PointNet classifier was trained on ModelNet, and [290] occasionally generates shapes that only rarely appear in ModelNet. In conjunction with their higher tendency for mode collapse, this partially accounts for their lower classification scores.

Metric	A	B	C	D	E	F
JSD	0.1660	0.1705	0.0372	0.0188	0.0077	**0.0048**
MMD-CD	0.0017	0.0042	0.0015	0.0018	0.0015	**0.0014**
CLF	84.10	87.00	96.10	94.53	89.35	87.40

Table A.7: Evaluating six generators on **train-split** of *chair* dataset on epochs/models selected via minimal JSD on the validation-split. We report: A: r-GAN, B: [290] (a volumetric approach), C: l-GAN(AE-CD), D: l-GAN(AE-EMD), E: l-WGAN(AE-EMD), F: GMM(AE-EMD). Note that the average classification score attained by the ground-truth point clouds was 84.7%.

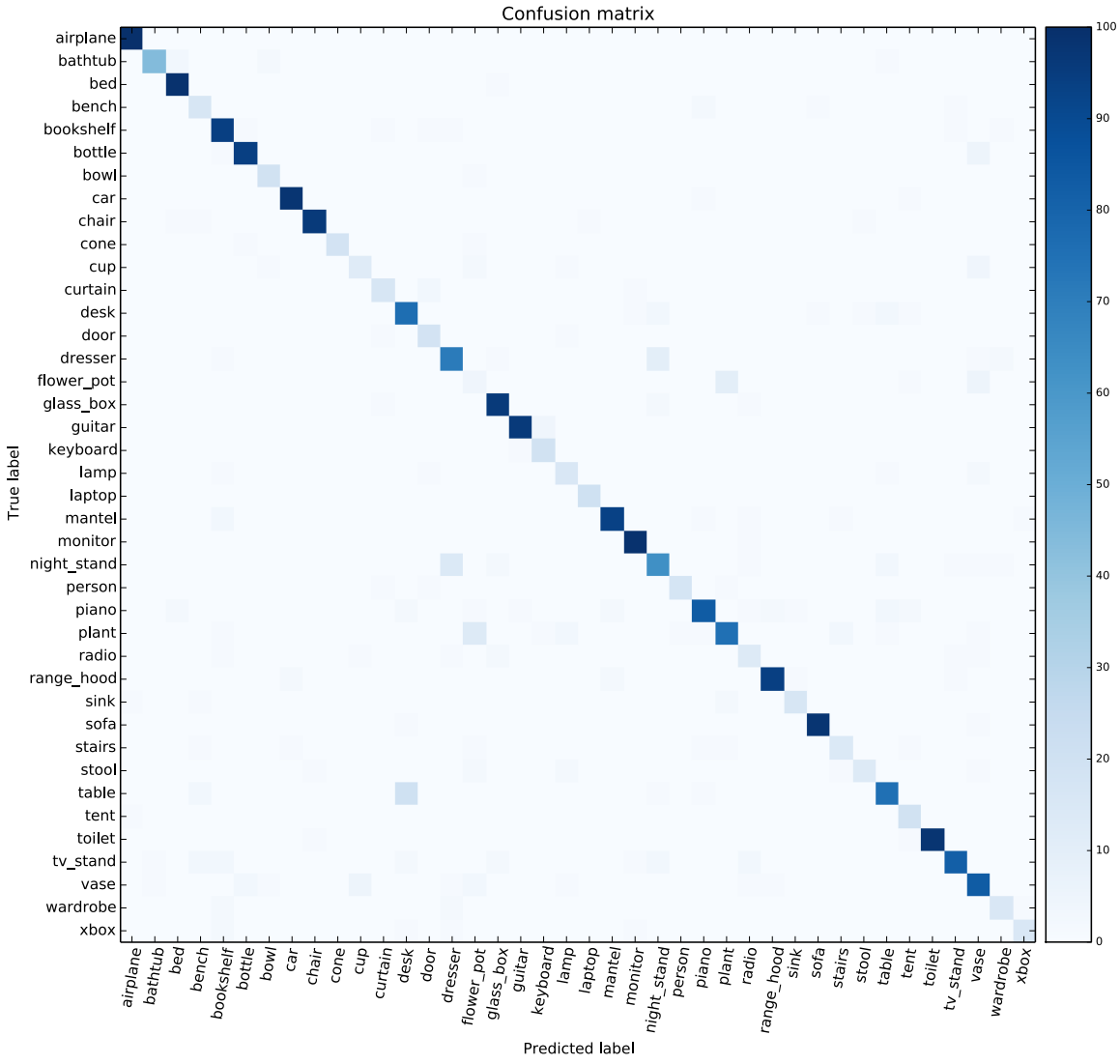


Figure A.2: Confusion matrix for the SVM-based classification of Section 5.1, for the Chamfer loss on ModelNet40. The class pairs most confused by the classifier are dresser/nightstand, flower pot/plant. Better viewed in the electronic version.

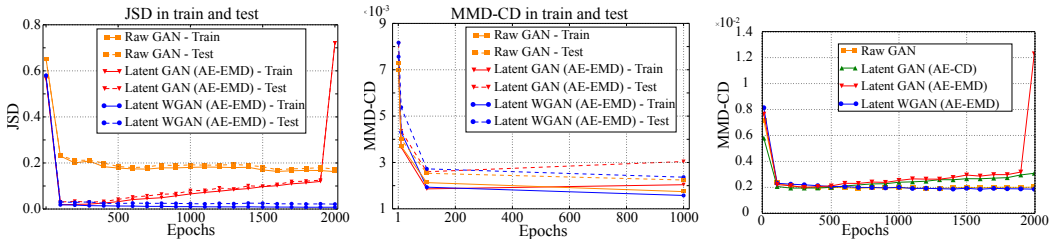


Figure A.3: Left/middle: Generalization error of the various GAN models, at various training epochs. Generalization is estimated using the JSD (left) and MMD-CD (middle) metrics, which measure closeness of the synthetic results to the training resp. test ground truth distributions. The plots show the measurements of various GANs. Right: Training trends in terms of the MMD-CD metric for the various GANs. Here, we sample a set of synthetic point clouds for each model, of size 3x the size of the ground truth test dataset, and measure how well this synthetic dataset matches the ground truth in terms of MMD-CD. This plot complements Fig. 2.11 (left), where a different evaluation measure was used - note the similar behavior.

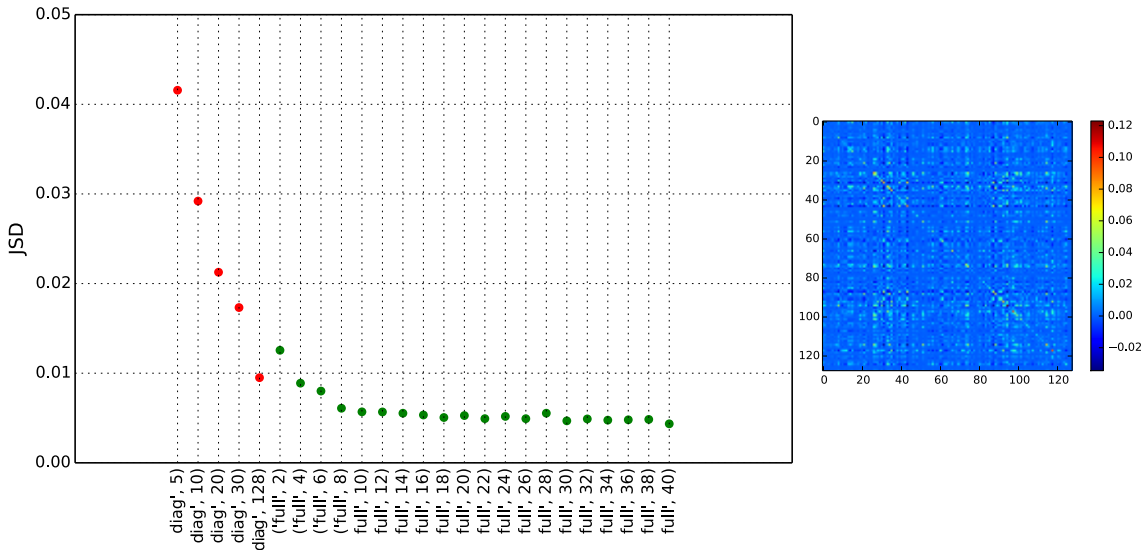


Figure A.4: GMM model selection. GMMs with a varying number of Gaussians and covariance type are trained on the latent space learned by and AE trained with EMD and a bottleneck of 128. Models with a full covariance matrix achieve significantly smaller JSD than models trained with diagonal covariance. For those with full covariance, 30 or more clusters seem sufficient to achieve minimal JSD. On the right, the values in a typical covariance matrix of a Gaussian component are shown in pseudocolor - note the strong off-diagonal components.

Figure A.5: The 32 centers of the GMM fitted to the latent codes, and decoded using the decoder of the AE-EMD.

Figure A.6: Point cloud *completions* of a network trained with partial and complete (input/output) point clouds and the **CD loss**. Each triplet shows the partial input from the test split (left-most), followed by the network's output (middle) and the complete ground-truth (right-most). Also compare with Fig 2.8 of thesis that portrays the corresponding completions of a network trained with the EMD loss.

Appendix B

Part-Aware Construction of Latent Spaces

B.0.1 Decomposer-Composer architecture details

The Decomposer consists of a whole-shape encoder and K projection layers, where K is the number of semantic part labels. The architecture of the whole-shape encoder is given in Table B.1. The projection layers are implemented as fully connected layers, with 100 outputs, where 100 is the dimension of the embedding space.

The Composer consists of a shared part decoder, and a Spatial Transformer Network (STN). The architecture of the part decoder is given in Table B.2. STN, similar to the original design in [115], consists of a localization sub-network, and a re-sampling module. The re-sampling module uses trilinear interpolation, and does not have learned parameters. The localization network receives both K stacked decoded parts, and the sum of part embeddings, of dimension 100. First, the two inputs are separately processed: the stacked decoded parts - using two FC layers with 256 outputs; the sum of part encodings - using one FC layer with 128 outputs. The two results are then concatenated into a single 384-dimensional vector, and are processed with two additional FC layers with 128 and 12 K outputs (K times 12 affine transformation parameters), respectively. All FC layers, except for the last one, are followed by ReLU layers, and dropout layers with a keep probability of 0.7.

B.0.2 Fine-grained classifier architecture details

In the evaluation of the proposed method, we used a binary classifier to estimate the quality of assembly and how "real" the resulting shapes looked (see Section 4.4 in the paper for detail). The

Type	Kernel	Stride	Outputs	Output size
conv.	$5 \times 5 \times 5$	$1 \times 1 \times 1$	16	32^3
conv.	$5 \times 5 \times 5$	$2 \times 2 \times 2$	32	16^3
conv.	$5 \times 5 \times 5$	$2 \times 2 \times 2$	64	8^3
conv.	$3 \times 3 \times 3$	$2 \times 2 \times 2$	128	4^3
conv.	$3 \times 3 \times 3$	$2 \times 2 \times 2$	256	2^3
FC	-	-	100	1

Table B.1: Whole-shape encoder (Decomposer) architecture. Each convolution layer ("conv.") is followed by a Rectified Linear Unit (ReLU) layer, and a batch normalization layer. The last is a fully-connected layer ("FC").

Type	Kernel	Stride	Outputs	Output size
FC	-	-	256	2^3
deconv.	$3 \times 3 \times 3$	$2 \times 2 \times 2$	128	4^3
deconv.	$3 \times 3 \times 3$	$2 \times 2 \times 2$	64	8^3
deconv.	$5 \times 5 \times 5$	$2 \times 2 \times 2$	32	16^3
deconv.	$5 \times 5 \times 5$	$2 \times 2 \times 2$	16	16^3
conv.	$5 \times 5 \times 5$	$1 \times 1 \times 1$	1	32^3

Table B.2: Part decoder (Composer) architecture. The fully-connected layer ("FC"), and every de-convolution layer ("deconv."), are followed by a Rectified Linear Unit (ReLU) layer, a batch normalization layer, and a dropout with keep probability 0.8.

architecture of the classifier is shown in Table B.3.

B.0.3 Additional shape classes

In this section we present shape reconstruction and manipulation results for two additional shape classes: airplanes and tables. Note that also here all the experiments were performed using unlabeled input shapes.

Figure B.1 presents the results of a single part exchange between two airplane. Figure B.2 presents the results of assembling an airplane shape from random parts. Figure B.3 presents the results of a single part exchange between two tables. The results demonstrate the ability of the proposed method to correctly place and scale the parts. Also, it presents an example (fifth table from the left) of a failure case of our algorithm on a challenging shape with a one-voxel wide legs, when it is unable to reconstruct the legs at all, but otherwise places the parts correctly.

Type	Kernel	Stride	Outputs	Output size
DO (0.5)	-	-	1	32^3
conv.	$6 \times 6 \times 6$	$2 \times 2 \times 2$	32	16^3
conv.	$6 \times 6 \times 6$	$2 \times 2 \times 2$	32	8^3
conv.	$4 \times 4 \times 4$	$2 \times 2 \times 2$	64	4^3
conv.	$2 \times 2 \times 2$	$2 \times 2 \times 2$	64	2^3
conv.	$2 \times 2 \times 2$	$2 \times 2 \times 2$	128	1
DO (0.5)	-	-	128	1
FC_1	-	-	128	1
FC_2	-	-	64	1
FC_3	-	-	2	1

Table B.3: Architecture of the binary classifier. Each convolution layer ("conv.") is followed by a Rectified Linear Unit (ReLU) and a batch normalization layers. Dropout layers ("DO") have keep probability of 0.5. The fully-connected layers FC_1 and FC_2 are followed by batch normalization and ReLU layers. The classifier produces binary output.

B.0.4 Miscellanea

Projection matrices

As illustrated in Figure B.4, the proposed method succeeds to obtain a set of projection matrices that sum to an identity. While these matrices are full rank, the plot of their singular values shows that their approximated ranks are in fact much lower.

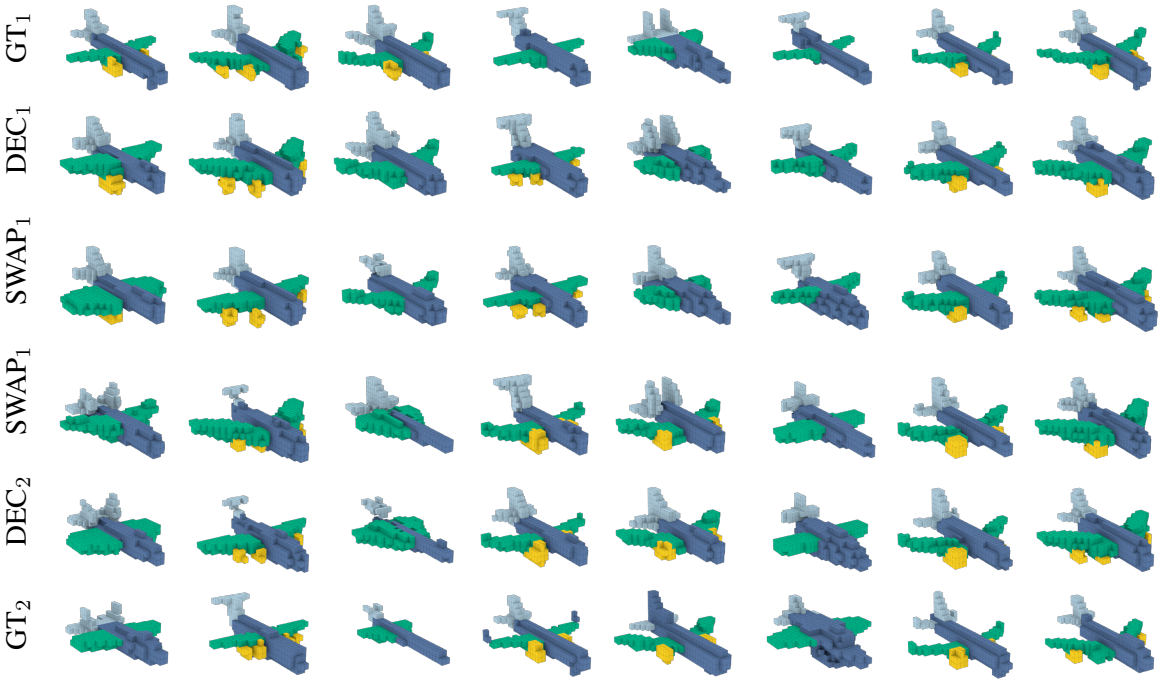


Figure B.1: Single part exchange experiment on airplane shapes. GT denotes ground truth labeled shapes, REC - reconstruction results, and SWAP - result of exchanging a part between the shapes.

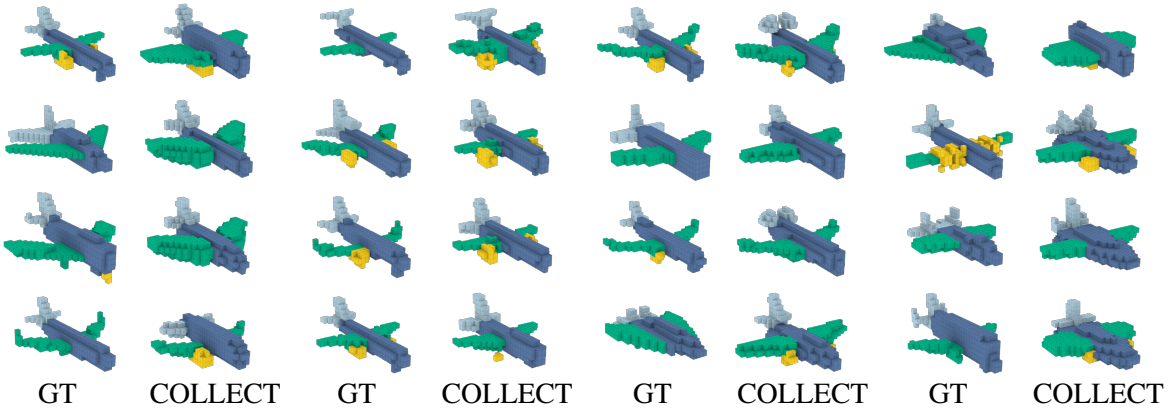


Figure B.2: Synthesis-from-parts example for aiplane shapes. GT denotes original test-set shapes, COLLECT - new shapes assembled from parts of the original shapes.

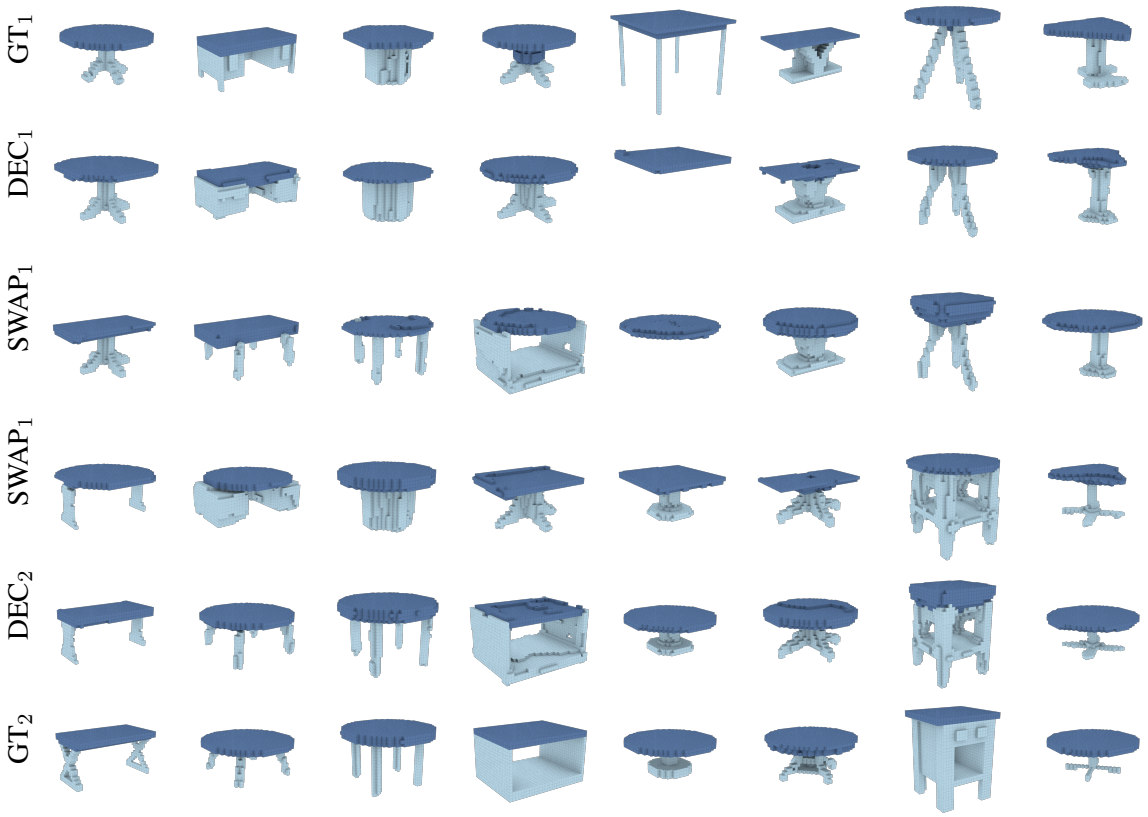


Figure B.3: Single part exchange experiment on table shapes. GT denotes ground truth labeled shapes, REC - reconstruction results, and SWAP - result of exchanging a part between the shapes (only two parts in this case).

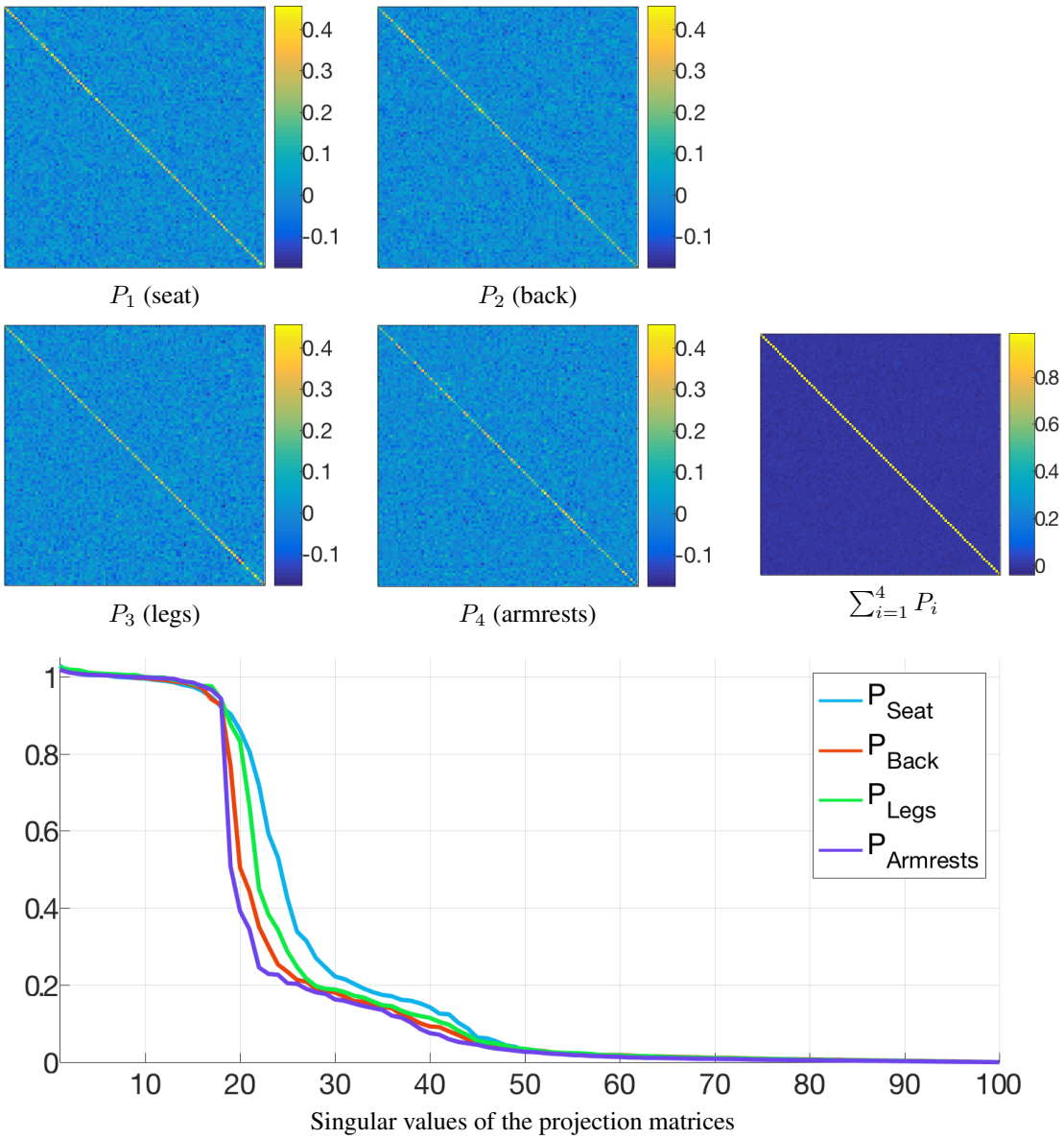


Figure B.4: Projection matrix analysis. Two upper rows present the obtained projection matrices, and their sum. The bottom row shows the singular values of the matrices.

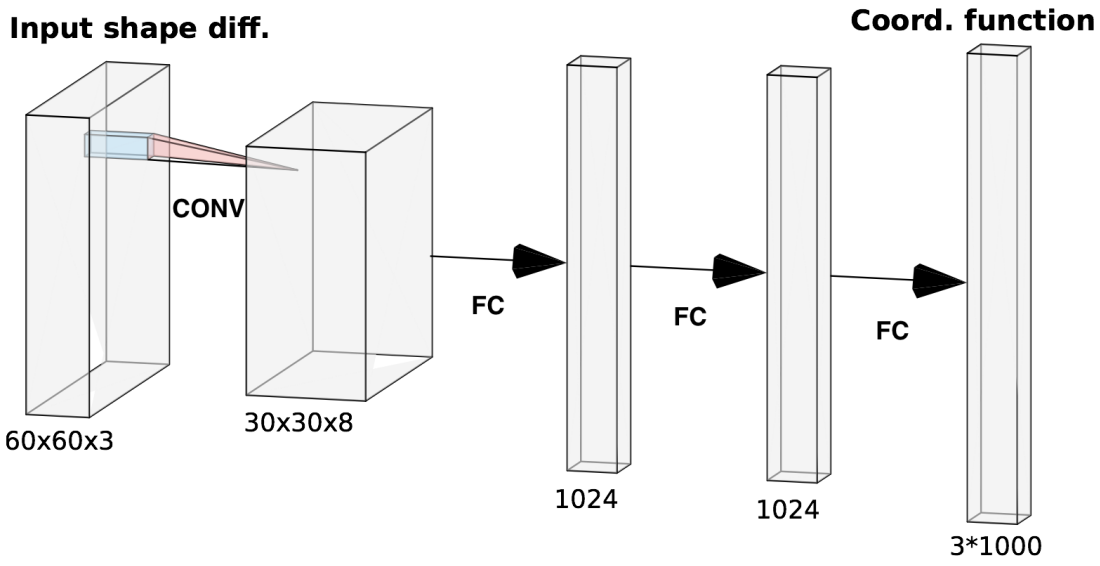


Figure B.5: OperatorNet architecture. Given the shape difference operators as input, OperatorNet outputs the coordinate functions of a shape. In particular, one can efficiently extract information from the difference operators (here considered as channels) with a simple and standard network architecture, which consists of a convolutional encoder and a fully connected decoder built with dense layers, as shown above.

B.1 OperatorNet

B.1.1 Proof of Theorem 1

Proof. Since X is known to be of rank 3, and $\mathbf{G}$ is symmetric, we have, by SVD:

$$\mathbf{G} = \mathbf{\Phi}^T A X X^T A \mathbf{\Phi} = U \Sigma U^T,$$

where, U, Σ are respectively the top 3 singular vectors and singular values of E^G. Therefore, we have $\mathbf{\Phi}^T A X R = U\sqrt{\Sigma}$, where R is a 3×3 rigid transformation matrix satisfying $R^T R = I_{3\times 3}$. In other words, we recover $\mathbf{\Phi}^T A \sim X$ from E^G, where $\sim X = XR$ is equivalent to X up to rigid transformations. And to recover the projection of $\sim X$ in span$(\mathbf{\Phi})$, we simply compute $\mathbf{\Phi} U\sqrt{\Sigma}$. □

Figure B.6: Top row: ground-truth embeddings; middle row: reconstructions from OperatorNet; bottom row: shapes from the training set, whose shape differences that are closest to the ones of the test shapes in the top row.

B.1.2 Verification of the Generalization power of OperatorNet

To demonstrate the generalization power of OperatorNet, we show in Figure B.6 our reconstructions of test shapes from the SURREAL dataset. For comparison, we also retrieve the shapes in the training set, whose shape differences are the nearest to the ones of the test shapes. In each of the figures, the top row presents the ground-truth test shapes; the middle row shows reconstructions from OperatorNet; the bottom row demonstrates the shapes retrieved from the training set via nearest neighbourhood search regarding shape differences.

It is evident that OperatorNet accurately reconstruct the test shapes, which deviate from the shapes in the training set significantly, suggesting that our network generalizes well in unseen data.

B.1.3 Comparison of Interpolation Schemes for Shape Differences

In the following experiment we note that, since the shape differences are represented by matrices, it is also possible to interpolate shape differences linearly, i.e., $\mathbf{D}(t) = (1-t)\mathbf{D}_0 + t\mathbf{D}_1$. However, as we argue in Section 4.2.1, the multiplicative property of shape differences suggests that it is more natural to interpolate the difference operators following Eq. (4.6). To illustrate this point, we show in Figure B.8 interpolated sequences with respect to the two schemes above – the multiplicative one in the top row and the linear one in the bottom row. It is visually evident that the former leads to more continuous and evenly deformed sequence. Moreover, we compute the distance between consecutive

shapes in both sequences and plot the distributions in the bottom panel of Figure B.7 as a quantitative verification.

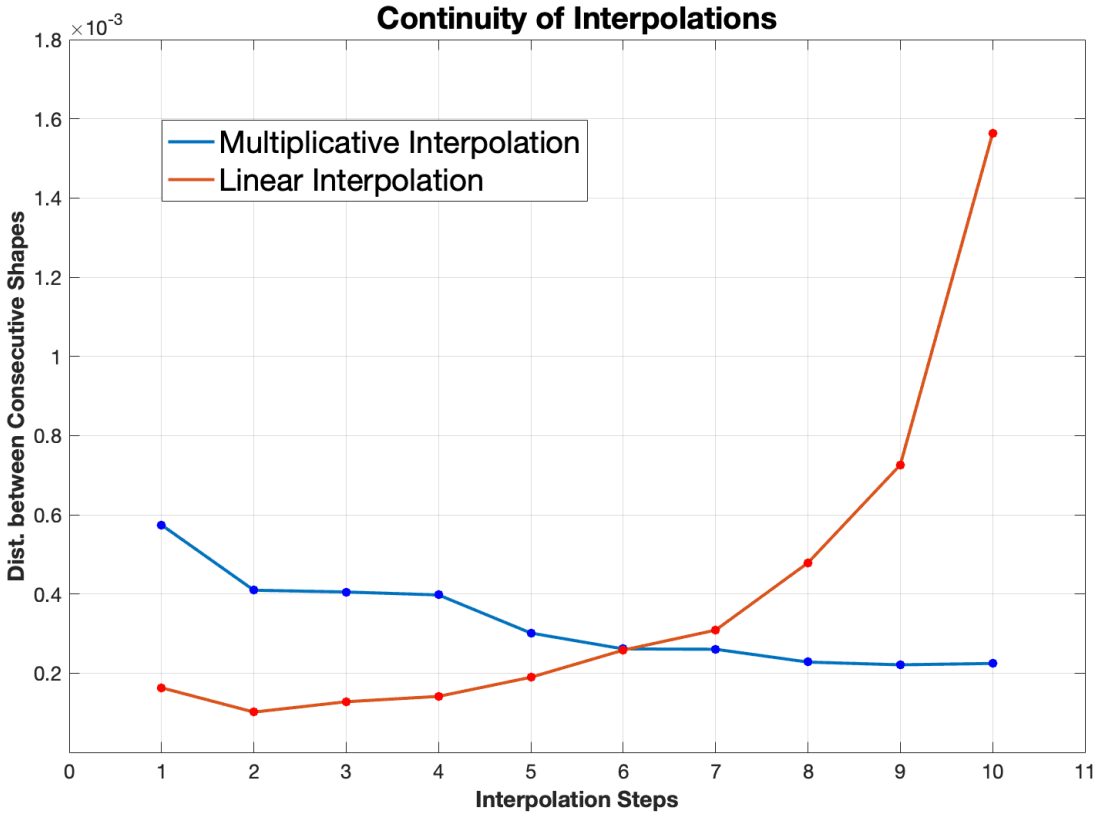


Figure B.7: The distances between consecutive reconstructed embeddings for both sequences. The multiplicative scheme clearly delivers more smooth deformation sequence.

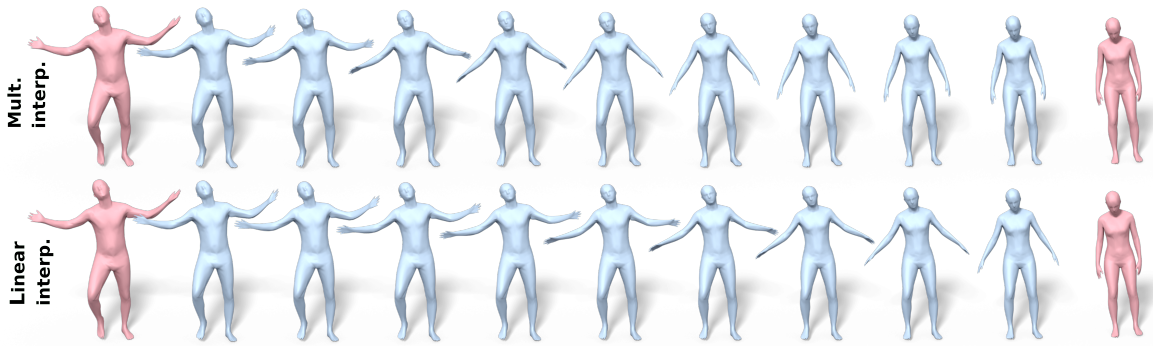


Figure B.8: Reconstructions regarding shape differences interpolated using multiplicative scheme (first row) and using linear scheme (second row).

B.1.4 Ablation Study on Network Design

We investigate multiple architectures for OperatorNet. In Table B.4 we compare the reconstruction performance over different combinations of input shape differences, and different depths of encoders.

We report the performance of 4 different convolutional encoders from 1 to 4 layers deep by doubling the number of neurons every layer.

Two trends are observed in Table B.4: first, we always achieve the best performance when all three types of shape differences are used, for all depth of the network; second, fixing the combination of input shape differences, the network performs better as its depth gets shallower.

Putting these two observations together, we justify our final model, which has one single layer convolutional encoder and uses all three types of shape differences as input.

Table B.4: Ablation study: auto-encoder performance on DFAUST testset (measured by the loss function as defined in Eq. (4.9), the errors in the table are at the scale of 10^{-4}).

Encoder architecture	Area	Ext	Conf	A+E	A+C	E+C	A+E+C
Conv. 8	**8.61**	**4.29**	**3.78**	**3.82**	**3.41**	**2.56**	**2.46**
Conv. 8×16	9.08	4.54	4.28	4.65	3.93	3.10	**3.05**
Conv. 8×16×32	9.90	5.54	4.91	5.59	4.88	3.71	**3.55**
Conv. 8×16×32×64	11.16	6.39	5.93	6.89	5.42	4.35	**4.24**

Appendix C

Discriminating the Shape of Objects with Referential Language

C.1 CiC details

To build the triplets comprising the communication contexts of ***CiC***, we exploited the *latent* (bottleneck-derived) vector space of a Point-Cloud based AutoEncoder (PC-AE) [6], trained with chair-only objects of ShapeNet [46]. Concretely, we used a PC-AE with small bottleneck (64D) to promote meaningful euclidean distances and after embedding all ~ 7000 ShapeNet chairs in the resulting space, we computed their underlying 2-(euclidean)-nearest-neighbor graph. On this graph, we selected the $1K$ chairs with the highest in-degree to 'seed' the triplet generation. For each of the 1K (seed) chairs, we considered it together with its two nearest neighbors from the *entire* shape collection to form a *Hard* triplet. Also, we considered it together with the two chairs that were closest to it but which were also more distant from it than the median of all pairwise distances, to form an *Easy* triplet. The above procedure gives rise to 2000 communication contexts when target vs. distractor information is ignored. However, to counterbalance the dataset while annotating these contexts in AMT, we ensured that *each* chair of a context was considered as a distractor and as a target, and that each resulting combination was annotated by at least 4 humans. Last, we note that when building the Hard triplets, we applied a manually tuned distance-threshold, to reject triplets that contained objects that were 'too' close: we found that about $\sim 3\%$ of chairs had a geometric duplicate that could vary only wrt. its texture.

C.2 Image and point-cloud pre-training

For the listeners and speakers we trained a PC-AE under the Chamfer loss [6] with a 128D bottleneck and point clouds with 2048 points extracted from 3D CAD models, uniformly area-wise. We also fine-tuned a VGG-16 pre-trained on ImageNet on a 8-way classification, with 36,632 rendered images of textureless 3D CAD models, taken from a single view-point. Concretely, we used images of the 8 largest object classes of Shape-Net (car, airplane, vessel, sofa, chair, table, lamp, riffle) and a uniformly random i.i.d. split of [90%, 5%, 5%] for train/test/val purposes. We fine-tuned the network for 30 epochs. During the first 15 epochs we optimized *only* the weights of the last (fc8) layer and during the last 15 epochs the weights of all layers. The attained test classification accuracy was 96.9%. Last, to embed an image for the downstream listening/speaking tasks, we used the 4096D output activations of the penultimate (fc7) fully-connected layer.

C.3 Pre-processing utterances

We preprocessed the collected human utterances by i) lowercasing, ii) tokenizing by splitting off punctuation, iii) tokenizing by splitting superlative or comparative adjectives ending in -er, -est to their stem word, e.g. 'thinner:' $\rightarrow$ ['thin', 'er'] and, iv) replacing tokens that appear once or not at all in a training split with a special symbol marking an unknown token (`<UNK>`). Furthermore, we ignored the utterances comprised by more than 33 tokens (99th percentile) and those for which the human listener in the underlying trial did not guess correctly the target. Last, we concatenated listener and speaker utterances from the same trial (in their order of formulation) by adding in the end of each but the last utterance a special symbol marking a dialogue: (`<DIA>`), e.g. ['the', 'thin', 'chair', `<DIA>`, 'yes'].

Architecture / Hyper Parameters	*Baseline*	*Early-Context*	*Combined-Interpretation*
Learning rate	0.0005	0.001	0.001
Label-smoothing	0.9	0.9	0.9
L_2 regularization	0.3	0.05	0.09
LSTM-input-dropout	0.5	0.7	0.45

Table C.1: Optimal hyper-parameters for ablated neural listener architectures, using both geometric modalities and word-attention and various degrees of context. Dropout numbers reflect the *keep* probability.

C.4 Listeners details

For the listeners we used a uni-directional LSTM cell with 100 hidden units, the output of which was passed into a 3-layer MLP with [100, 50, 3] neurons that predicted the triplet's classification logits. To the output of each hidden layer of the MLP, batch normalization [113] and a ReLU [173] non-linearity was applied. The listeners' word-embedding was initialized with a $100D$ GloVe embedding pre-trained on the 6B Wikipedia 2014 corpus, and which was further fine-tuned during training. The PC-AE (128D) and VGG (4096D) latent vectors, that encoded each object, were passed as *input* to the LSTM when only one geometric modality was used. When the two modalities used together, the PC-AE codes were concatenated with the *output* of the LSTM, and the concatenated result was processed by the final MLP. In either case, we first re-embedded these geometric codes (100D) with 2 separate/single FC-ReLU layers (referred as 'projection' layers in the Section 5.5. An overview of the proposed listener reflecting the overall design choices is given in Fig.5.3. We used dropout with 0.5 keep probability *before* the 'projection' layers with a drop-out mask that was the same for the objects of a given triplet. Separate dropout with 0.5 keep probability was applied in all input vectors of the LSTM (i.e. on the language tokens or the grounding geometric codes). Last, the ground-truth indicator vectors of each triplet were label-smoothed [264] by assigning 0.933 probability mass to the target and 0.0333 to the distractors (i.e. smoothing of 0.9).

Discussion Label smoothing yielded a mild performance boost of $\sim 2\%$ across all ablated listener architectures, in accordance with previous work [264]. We note that we did not manage to improve the best attained accuracies by applying layer normalization [24] in the LSTM, or adversarial regularization [189] on the word-embedding. Dropout [257] was by far the most effective form of regularization for our listeners ($\sim$[8-9]%), following by L_2 weight-regularization of the projection layers ($\sim$[2-3]%). Finally, using a separate MLP to process the PC-AE codes, was slightly better than feeding them directly in the LSTM (after the tokens of each utterance were processed). However, grounding the LSTM with the PC-AE codes, and using the VGG codes in the end of the pipeline (either via pre-MLP concatenation or by feeding the latter in the LSTM) deteriorate *significantly* all attained results.

Context Ablations We ablated three architectures that used simultaneously images and point-clouds, word attention and different degrees of context (see Section 5.5 in the thesis). The optimal Hyper-Parameters (HP) for each architecture are shown in Table C.1. We did a grid search over the space of HP associated with each architecture *separately*. To circumvent the exponential growth of

this space, we search it into two phases. First, we optimized the learning rate (in the regime of [0.0001, 0.0005, 0.001, 0.002, 0.004, 0.005]) in conjunction with the drop-out (keep probability) applied at the LSTM's *input*, in the range [0.4-0.7] with increments of 0.05. Given the acquired optimal values, we searched for the optimal L_2 weight-regularization (in the range of [0.005, 0.01, 0.05, 0.1, 0.3, 0.9]) applied at the two projection layers, and label-smoothing ([0.8, 0.9, 1.0]). For these experiments we used a single random seed to control for the data splits with the *object-generalization task*. We note that for the *Early-Context* listener, using a single 1D convolutional layer to extract the grounding vector of each object, appeared to produce better results than using a single FC layer (or deeper alternatives). This single convolutional layer we used, converted the input signal $[f(v_j, v_k)||g(v_j, v_k)||v_i] \in \mathbb{R}^{100\times 3}$ to a $\mathbb{R}^{100\times 1}$ LSTM-grounding vector for *e*ach object v_i, with an $8 \times 3 \times 1$ kernel and stride 1.

Training We trained the *Baseline* and the *Combined-Interpretation* for 500 epochs and the *Early-Context* for 350. This was sufficient, as more training increased overfitting without improving the attained test/val accuracies. We halved the learning every 50 epochs, if the validation error was not improved in any of them. Namely, every 5 epochs we evaluated the model on the validation split in order to select the epoch/weights with the best accuracy. Because the *Combined-Interpretation* is sensitive in the input order of the object codes, we randomly permute them during training. We use the Adam [129] ($\beta_1 = 0.9$) optimizer for all experiments.

C.5 Speaker details

Image-based speaker To find good model parameters for an image-based speaker, we considered a hyper-parameter search on a *literal* variant. Similarly, to what we did in the ablations of listener variants we conducted a two-stage grid search given a single random seed and the *object generalization* task. At the first stage, we searched models varying: a) the hidden neurons of the LSTM (100 or 200), b) the initial learning rate ([0.0005, 0.001, 0.003]), c) the drop-out keep probability applied on the word-embeddings ([0.8, 0.9, 1.0]) and d) the dropout keep probability applied at the LSTM's output ([0.8, 0.9, 1.0]). The two best performing models were further optimized by considering L_2-weight regularization applied at the FC-projection layer (with values in [0, 0.005, 0.01]) and the dropout keep-probability applied before the FC-projection layer ([0.5, 0.7, 0.9 1.0]). The resulting optimal parameters are reported in Table C.2.

LSTM Size	Learning rate	L_2-reg.	Word-Dropout	Image-Dropout	LSTM-out Dropout
200	0.003	0.005	0.8	0.5	0.9

Table C.2: Optimal hyper parameters for *literal* image-based neural-speaker. The dropout numbers reflect keep probabilities and the Image-Dropout refers to the dropout applied at the VGG-image codes, before the FC-projection layer.

Point-cloud-based speaker For the point-based speaker, we did a similar but more constrained hyper-parameter search as we did for the image-based speaker, by also considering its *literal* variant. Here, we fixed the drop-out applied the word-embeddings and to the LSTM's output (0.8 and 0.9 keep-probability respectively) and ablated the remaining hyper-parameters as we did for the image-based speaker. We found the same configuration of parameters (Table C.2) to be optimal for point-based models as well. Exception to this was the the dropout applied to the PC-AE codes before the FC-projection (no dropout at all was best in this case). Also, the point-based speakers needed more training to converge than the image-based ones (maximally 400 epochs vs. 300).

Model selection To do model selection for a training speaker, we used a pre-trained listener (with the same train/test/val splits) which evaluated the synthetic utterances produced by the speaker during training. To this purpose the speaker generated 1 utterance for each unique triplet in the validation set via greedy (arg-max) sampling every 10 epochs of training and the listener reported the accuracy of predicting the target given the synthetic utterance. In the end of training (300 epochs for image-based speakers vs. 400 for point-based ones), the epoch/model with the highest accuracy was selected.

Other details We initially used GloVe to provide our speakers pre-trained word embeddings, as in the listener, but found that it was sufficient to train the word embedding from uniformly random initialized weights (we used the range [-0.1, 0.1]). We also initialized the bias terms of the linear word-encoding layer with the log probability of the frequency of each word in the training data [122], which provided faster convergence. We train with SGD and Adam ($\beta_1 = 0.9$) and apply norm-wise gradient clipping with a cut-off threshold of 5.0. The training utterances have a maximal length of 33 tokens (99th percentile of the dataset). For any speaker we sampled utterances of the maximum training length. For the *pragmatic* speaker we sample and score 50 utterances per triplet at test time (following Eq. 5.1 of the thesis).

Point-cloud & image-based speaker In *preliminary* experiments, we attempted to incorporate both geometric modalities: point-clouds and images in a speaker network, similarly to what we did

Approach	Listener's Accuracy
Concat (100D)	$65.1 \pm 0.51\%$
Concat (200D)	$78.2 \pm 0.95\%$
Sum	$77.9 \pm 0.38\%$
Serial	$\mathbf{79.0} \pm 0.32\%$

Table C.3: Ablating approaches for incorporating simultaneously point-clouds with images in a *literal* neural-speakers. *Sum*: Summing the two latent codes for each object. *Concat*: Concatenating the codes. *Serial*: Feeding them one after the other in the LSTM. Concatenation naturally doubles the input-dimensions of the LSTM (Concat 200D). To keep them the same as with all other experiments (100D) we also tested reducing the VGG/PC-AE projection layers to 50 dimensions for each modality (Concat 100D). Results are averages of 5 samples of utterances for a fixed test dataset.

for the best-performing listener. While, this resulted in a (*literal*) speaker model that could achieve higher neural-listener evaluation-accuracy than when either modality was used in isolation, we did not observe any improvement against the image-based speaker in AMT *human*-listener experiments.

We attempted three ways of 'mixing' the two modalities in a speaker. Namely, for each object of a communication context: a) providing the LSTM with the *concatenation* of its projected VGG code and its projected PC-AE code, b) same as a) but instead of concatenation, using the *sum* operator, c) first providing its PC-AE projected code followed at the *next time* step by its VGG one. We compared these approaches by using the optimal hyper-parameters for an image-based speaker and only vary the amount of dropout applied to the point-cloud before the projection layer ([1.0 0.8, 0.6] keep probability). In all cases, avoiding dropout was best. The final results for a single random-seed and the object-generalization task are reported in Table C.3. We note that while the optimal speaker that used two modalities performed slightly better than the image-based speaker, per neural-listener evaluation, it did not improve the attained performance in preliminary experiments with of human listeners in AMT.

C.6 Further quantitative results

C.6.1 Listeners: context incorporation

Architecture	Overall	Subpopulations				
		Hard	Easy	Sup-Comp	Negative	Split
Combined-Interpretation	$75.9 \pm 0.5\%$	$67.4 \pm 1.0\%$	$83.8 \pm 0.6\%$	$74.4 \pm 1.5\%$	$77.3 \pm 1.5\%$	$65.8 \pm 5.2\%$
Early-Context	$79.4 \pm 0.8\%$	**70.1** $\pm 1.3\%$	$88.1 \pm 0.6\%$	$75.6 \pm 2.2\%$	**78.9** $\pm 1.4\%$	**67.4** $\pm 3.6\%$
Baseline	**79.6** $\pm 0.8\%$	$69.9 \pm 1.3\%$	**88.8** $\pm 0.4\%$	**76.3** $\pm 1.3\%$	$77.5 \pm 1.2\%$	$62.5 \pm 3.7\%$

Table C.4: Comparing the effect of context inspection for listening on various (test) subpopulations of the *object generalization* task. The listeners use images, point-clouds and word-attention. Reporting averages of five random seeds controlling the split populations and the network's initialization.

Architecture	Overall	Subpopulations				
		Hard	Easy	Sup-Comp	Negative	Split
Combined-Interpretation	$78.4 \pm 0.2\%$	$71.5 \pm 0.6\%$	$85.2 \pm 0.3\%$	$75.8 \pm 0.9\%$	$77.6 \pm 0.8\%$	$61.8 \pm 3.0\%$
Early-Context	**84.4** $\pm 0.5\%$	**78.5** $\pm 0.8\%$	$90.2 \pm 0.7\%$	**80.9** $\pm 0.6\%$	**82.6** $\pm 1.1\%$	**68.9** $\pm 2.3\%$
Baseline	$83.7 \pm 0.2\%$	$77.0 \pm 0.8\%$	**90.3** $\pm 0.3\%$	$80.8 \pm 0.8\%$	$80.5 \pm 1.0\%$	$64.6 \pm 3.7\%$

Table C.5: Comparing the effect of context inspection for listening on various (test) subpopulations of the *language generalization* task. The listeners use images, point-clouds and word-attention. Reporting averages of five random seeds controlling the split populations and the network's initialization.

In Table C.4 we complement the results presented in the thesis at Table 5.1, by including two more sub-populations ('Negative' and 'Split'). In Table C.5, we repeat this study for listeners trained and tested on the *language generalization* task. 'Negative' is a subpopulation of utterances that contain at least one word of negative content e.g. 'not', 'but' etc. and is comprised by $\sim 15.0\%$ of all test utterances. 'Split' is smaller subpopulation ($\sim 3.2\%$ of test data) that includes language the explicitly contrasts the target with the distractors e.g. 'from the two that have thin legs, the one...'. We used an ad hoc set of search queries to find such utterances among the test set and found that the *Early-Context* architecture does perform noticeably better on these utterances. However, given the low occurrence of such cases, the resulting effects were not significant and we decided the gains of *Early-Context* architecture were not worth the increase in model complexity and rigidity with respect to context size.

C.6.2 Listeners: part-lesion

	Single Part Lesioned	**Single Part Present**
Mentioned Part	$44.9\% \pm 1.2$	$67.2\% \pm 1.1$
Random Part	$68.9\% \pm 1.3$	$42.3\% \pm 1.3$

Table C.6: Evaluating the part-awareness of neural listeners by lesioning object *parts*. Results shown are for listeners using **both** point-clouds and images, with average accuracy of 78.8% when *intact* objects are used.

We complement Table 5.3 of the thesis, with a similar study (Table C.6) where we ablate our neural listeners with regards to their sensitivity in referential utterances based on object parts, when *both* geometric modalities are used. We have observed that the PC-AE attempts to reconstruct (decode) noisy but *c*omplete models, even when the input is a partial, which could explain the gains seen in Table C.6 compared to Table 5.3 when lesioning parts.

C.6.3 Speakers: length penalty and listener awareness

To find the optimal length-penalty value (α, thesis Eq.1) for image-based *literal* and a *context-unaware* speaker variants, we used our best-performing listener to simultaneously score and evaluate the utterances produced by the speakers for different values of α (Fig. C.1). The best performing length penalty for a context-unaware speaker is 0.7, and for a literal 0.6. Given the optimal α values, for these models we show the effect of using different degrees of listener-awareness (β) in Fig. C.1.

	Population		
Class	entire	with part	without part
chair	7.1	8.0 (77%)	4.7 (21%)
bed	6.4	7.0 (26%)	5.3 (48%)
lamp	7.3	11.0 (20%)	5.9 (37%)
sofa	10.1	11.0 (72%)	5.9 (15%)
table	6.6	8.0 (40%)	4.9 (42%)
average	7.6	9.3 (39.5%)	5.5 (35.5%)

Table C.7: Average length of utterances for various transfer classes (complementing Table 5, Main Paper). Between parentheses is reported the percentage of the entire population that is captured by its specific sub-population. The average (last row) is wrt. the transfer classes only; the chair-category is displayed for reference.

It is interesting to observe that even the context-unaware speaker can generate utterances that an evaluating listener can find them very discriminative, as long as is allows to rank them.

In Fig. C.2 we demonstrate the effect that the relative (training) size of the evaluating listener vs. the 'internal' listener used by a *pragmatic* speaker has for the evaluating accuracy, for two values of β. In either case we observe a slow decline in evaluating accuracy as the training size for the evaluating listener increases (from 0.5 to 0.9) and consequently the training size for the 'internal' listener decreases (from 0.5 to 0.1).

Understanding out-of-class reference

We complement the Table 5.5 with the standard-deviations of the underlying accuracies in Table C.8. We also report simple statistics regarding the underlying transfer classes in Table C.7. We note that the transfer learning accuracies acquired by listeners operating with both point-clouds and images for these experiments were significantly lower ($\sim 7\%$ on average). We hypothesize that this is due to the fact that our (chair-trained) listener models that utilize point-clouds, rely on a pre-trained single-class PC-AE, unlike the pre-trained VGG (image encoder) which was fine-tuned with multiple ShapeNet classes. Also, for these experiments, $[\sim 1\% \sim 7\%]$ (depending on the transfer class) of the tokens were not in the chair-vocabulary, and we chose to ignore them i.e. treat them as white-space. Last, per Table C.7 in *all* transfer classes the *with-part* population contains quite larger utterances than the *without-part* (9.3 vs. 5.5 on average) and that even in the case of lamps, arguably the most dissimilar category from chairs, $20 + 37 = 57\%$ of the collected utterances are in the *known* population.

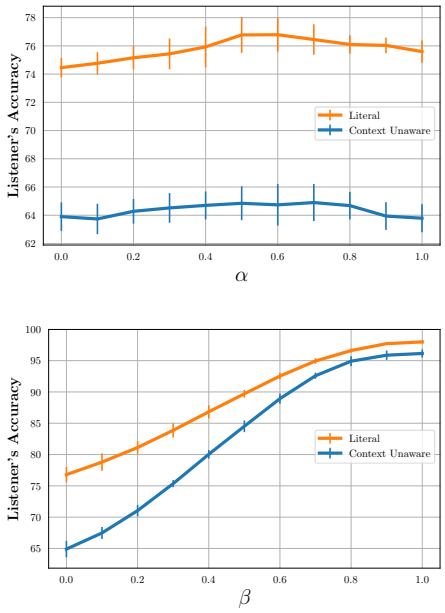


Figure C.1: Left: Measuring the effect of using different length-penalty (α) values to select the top-1 scoring utterance for context-unaware and pragmatic speakers for contexts of the *object generalization* validation split (left). Right, measuring the the effect of various β-values used in turning the context-unaware and literal speakers ($\beta = 0.0$) to *pragmatic* speakers, under the optimal α of the left figure. In both plots, the y-axis reflects the performance of a listener who is used to rank *and* evaluate the utterances. Averages are with respect to 5 random seeds controlling the data splits and the initializations of the neural-networks.

Population \ Class	bed	chair	lamp	sofa	table
entire	56.4 ±2.0%	77.4 ±0.9%	50.1 ±1.3%	53.6 ±2.0%	63.7 ±1.2%
known	55.8 ±1.5%	77.8 ±0.8%	51.9 ±1.8%	55.0 ±2.0%	65.5 ±0.9%
with part	**63.8** ±4.2%	77.0 ±0.8%	**60.3** ±4.4%	**55.1** ±2.5%	**68.3** ±2.6%
without part	51.5 ±3.0%	**80.5** ±1.2%	47.1 ±2.8%	54.7 ±5.5%	62.7 ±0.9%

Table C.8: Transfer-learning of neural listeners in novel object *classes*: average accuracies *with* standard deviations (complementing Table C.8, Main Paper). The sub-populations denote *entire*: all collected utterances, *known*: utterances containing *only* chair-training-vocabulary words, *with-part*: subset of *known*, with utterances containing at least one part-related word, *without-part* subset of *known* and complement of *with-part*. For reference the test-chair statistics are shown (first row) but not included in the reported average (last row).

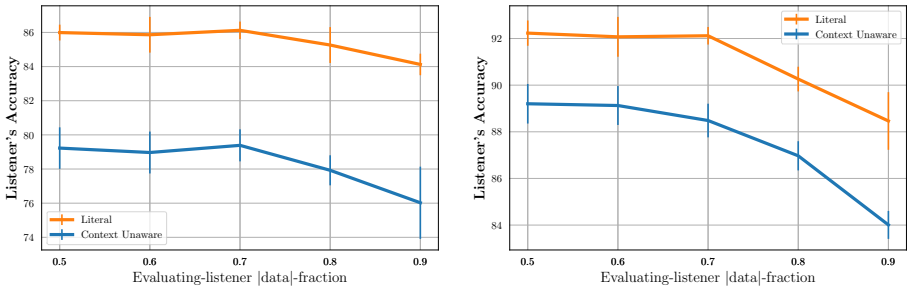


Figure C.2: Effect of partitioning the training data for the evaluating and 'internal' listeners. Here, we turn context-unaware and literal speakers into pragmatic ones under two β values. The x-axis shows the fraction (f) of the training data that was used to train the *evaluating* listener (the remaining $100 - f\%$ is used to train the *internal* listener) of the resulting pragmatic speaker. On the y-axis we display the performance of the evaluating listener for the top-scoring model-generated utterance. Left: Effect of the length-penalty. Right: Speakers using the most aggressive $\beta = 1.0$ value.

Figure C.3: **Examples of lesioning all but the mentioned part**. Here, we show the response of a *Baseline* listener tested with visual representations of entire objects (left column, three chairs) vs. its response when it receives **only** the visual features corresponding to the referred semantic-part (right column). The corresponding utterance is shown left-most of each row. In these examples the listener assigns higher confidence to the actual target when the isolated parts are considered instead of the entire objects, implying that further performance gains can occur with an explicit part-aware visual attention mechanism.

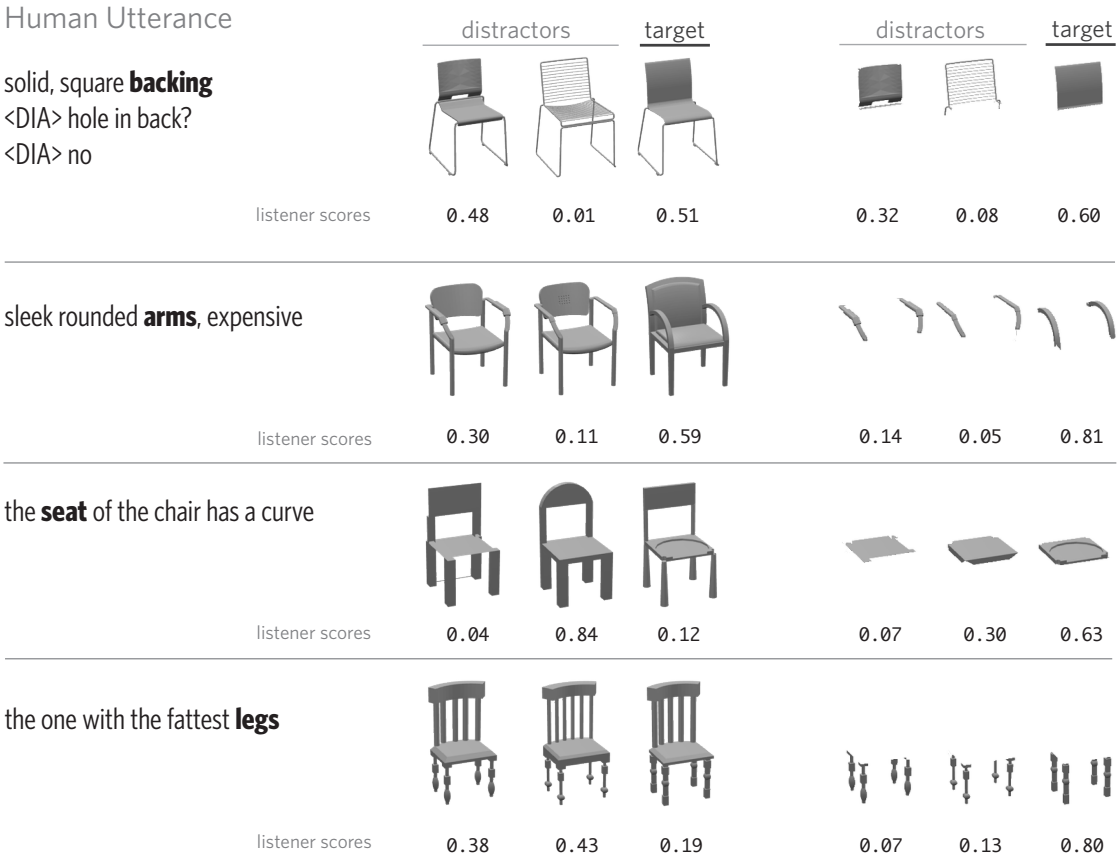


Figure C.4: **Pragmatic vs. literal speakers for two modalities**. More examples of pragmatic vs. literal generations in Hard contexts. Tor-row includes examples from image-based speakers. Bottom-row from point-based ones.

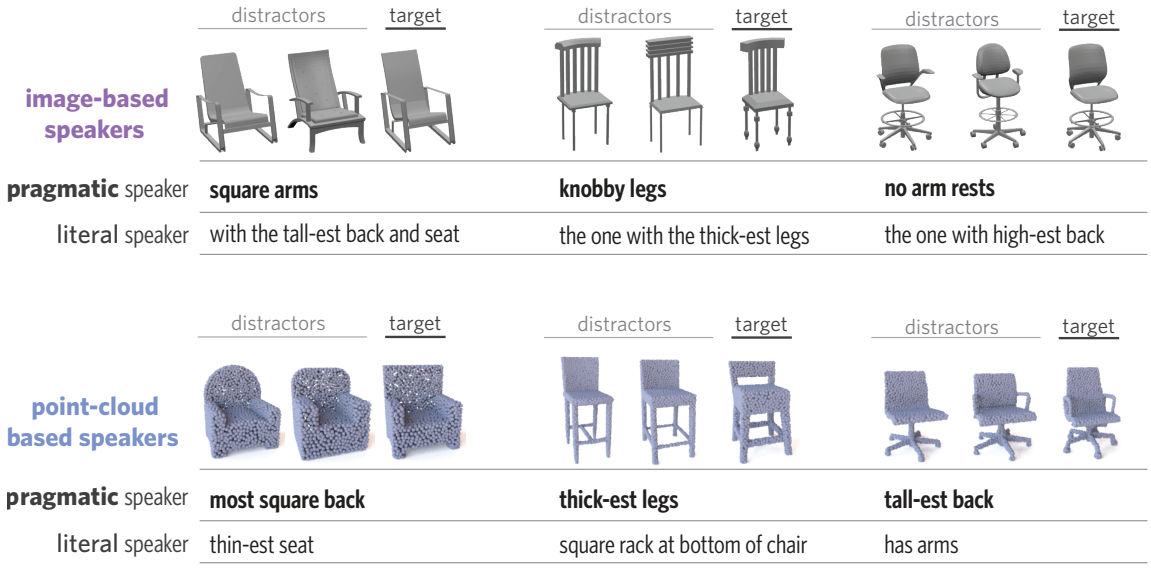


Figure C.5: **Speaking in *novel* classes.** In orange-color are *model*-generations of a chair-speaker describing the *non-chair* object above it (inside each orange box). Here, we first use a chair-listener to create easily-separable communication contexts to which we then apply our speaker. Concretely, the listener scores the utterance-object compatibility of *all* ShapeNet objects of a given class e.g. table, under a 'query' (utterance) e.g. 'modern'. We use the top-5 scoring and least-5 scoring objects (shown in left/right panels of each row respectively) to select at random a *target* (orange box) from the former set and two *distractors* (cyan boxes) from the latter. In the resulting communication context, we apply our speaker and display in orange-color the generation. The queries used for this experiment are shown in the left-most part of the Figure.

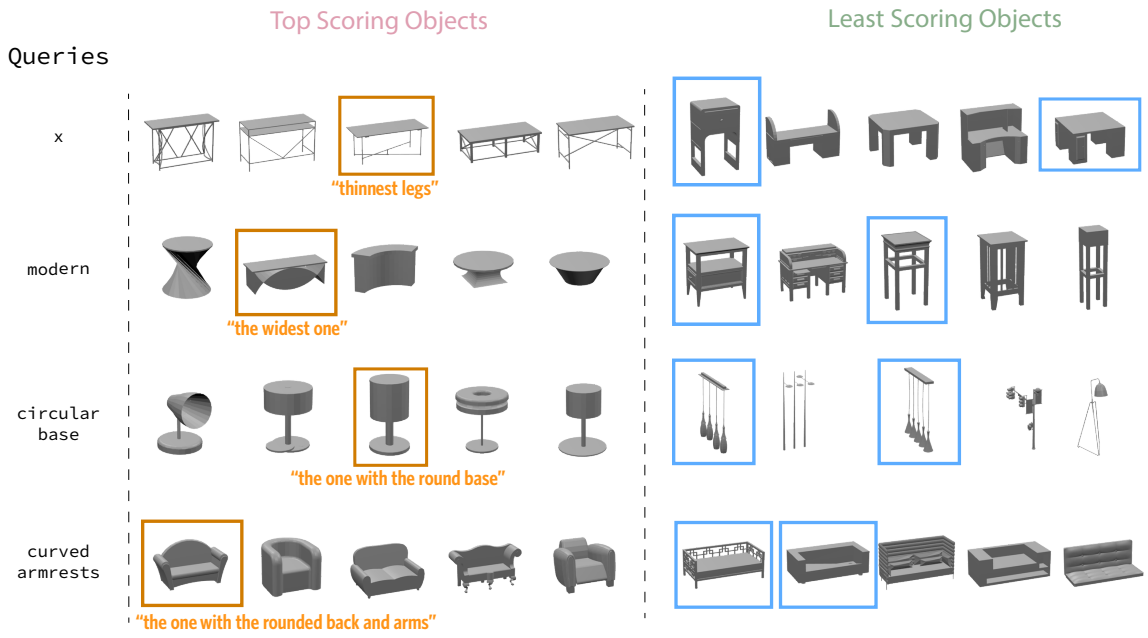
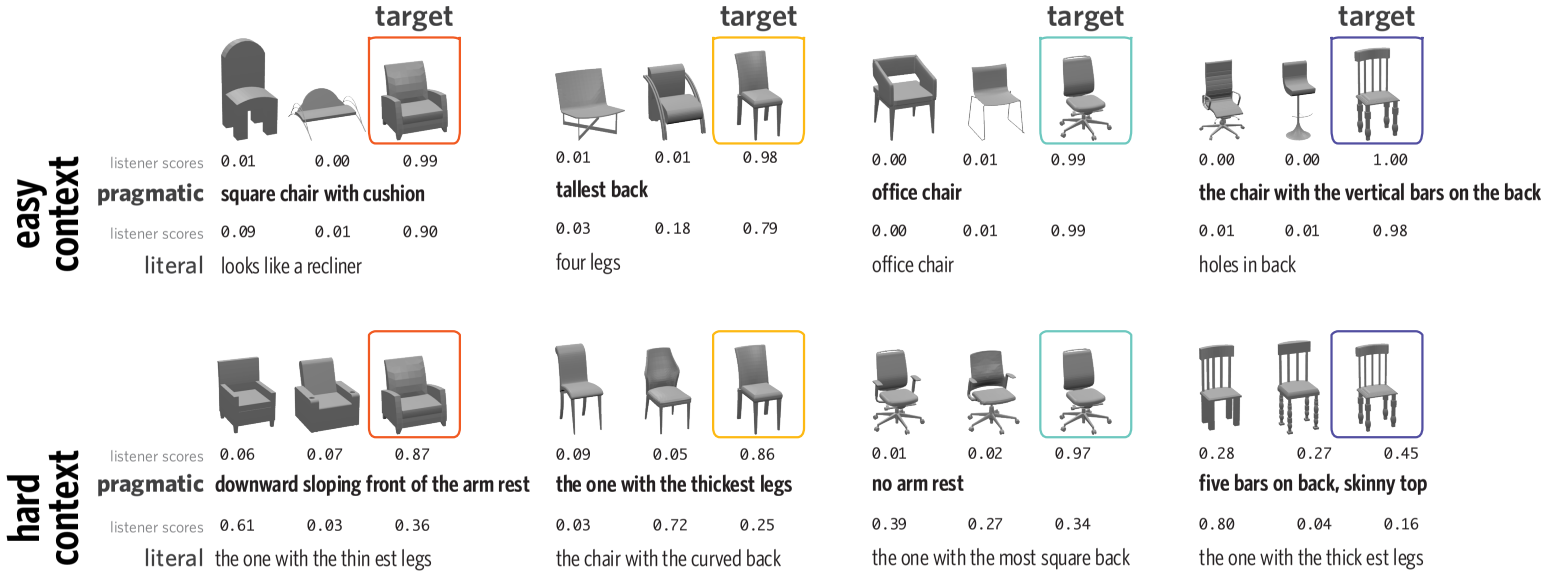


Figure C.6: **Effect of context on production**: Synthetic utterances generated by a *literal* and *pragmatic* image-based speaker. The top and bottom rows show utterances produced for the same target in a Easy and Hard context, respectively. The *Baseline* (with point-clouds and images and attention) listener is used to predict the target and its confidence is displayed above each utterance. While both speaker models produce similarly effective utterances in Easy contexts, the literal speaker fails to produce effective utterances in Hard contexts.



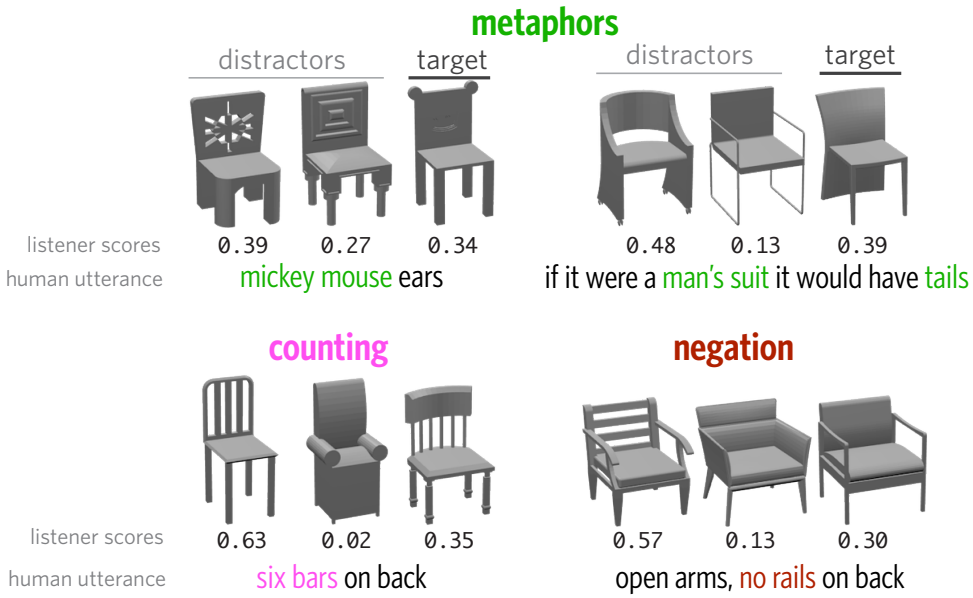


Figure C.7: **Neural-listener failure cases.** Our top-performing listener model appears to struggle to interpret referential language that relies on metaphors, precisely counting parts, or (to a less degree) negations. All examples are drawn from the test set and were correctly classified by human listeners in the original task.

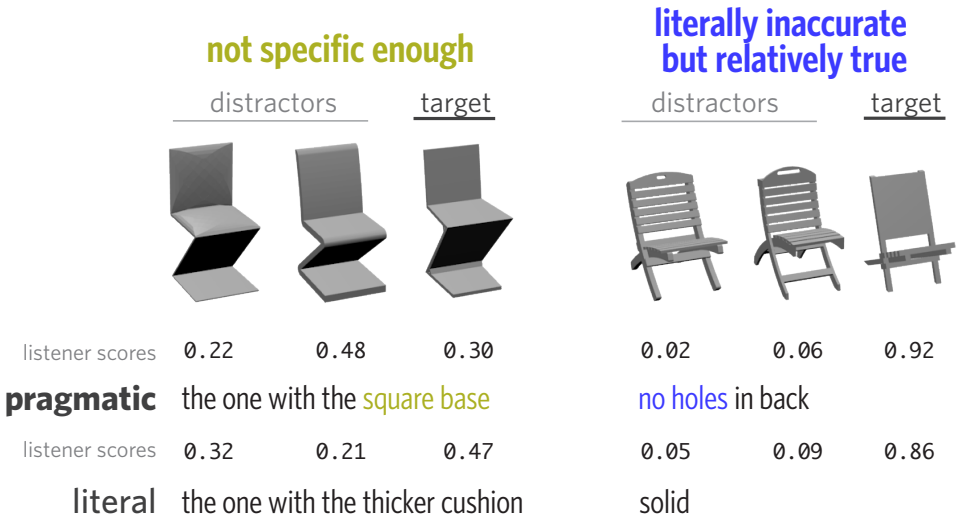


Figure C.8: **Neural-speaker failure cases.** Sometimes even the *pragmatic* speaker produces insufficiently specific utterances that mention only undiagnostic features, or produces utterances that are literally false of the target (e.g. there technically *is* a hole in the back) while still succeeding in distinguishing the objects.

C.7 Miscellaneous

Easy	**word**	office	sofa	regular	folding	wooden	stool	wheels	metal	normal	rocking
	pmi	-1.70	-0.94	-0.88	-0.84	-0.83	-0.79	-0.78	-0.71	-0.67	-0.66
Hard	**word**	alike	identical	thickness	texture	darker	skinnier	thicker	perfect	similar	larger
	pmi	0.69	0.67	0.67	0.66	0.65	0.64	0.63	0.62	0.62	0.61

Table C.9: Most distinctive words in each context type according to point-wise mutual information (excluding tokens that appeared fewer than 30 times in the dataset). Lower numbers are more distinctive of Easy and higher numbers are more distinctive of Hard.

Each game consisted of 69 trials (unique triplets) and participants swapped speaker and listener roles with the conclusion of each trial. The game's interface is depicted in Figure C.9. Participants were allowed to play multiple games, but most participants in our dataset played exactly one game (81% of participants). The most distinctive words in each triplet type (as measured by point-wise mutual information) are shown in Table C.9).

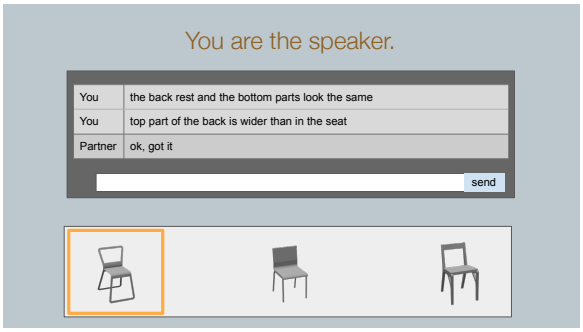


Figure C.9: Reference game interface. Communication was natural without any system constraints being imposed.

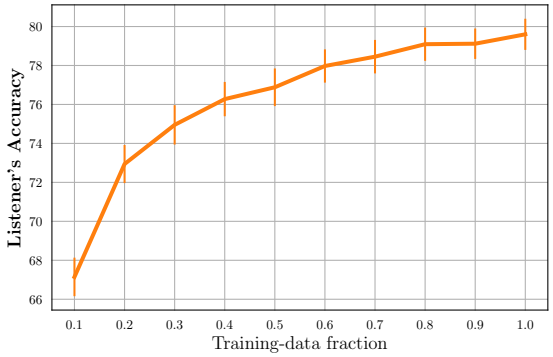


Figure C.10: Listener's accuracy for different sizes of training data, under the *object* generalization task. The original split includes [80%, 10%, 10%] for training/test/val purposes, thus the maximum size of training data is 0.8 of the entire dataset corresponding to the value (fraction) 1.0 in the x-axis. The listener model uses the *Baseline* architecture with word attention, images and point-clouds and its accuracy is measured on the original (10%) test split. Results are averages of 5 random seeds controlling the original data split and the neural-net's initialization.

Appendix D

Referential Language for Object Discrimination in the Real-World

D.1 Building Nr3D Details

D.1.1 Making Stimuli

Nr3D is comprised by a total of 41,503 utterances describing objects belonging in one 76 fine-grained object classes in 5,878 communication contexts (unique sets {S, I}, where S denotes a specific scene, and I the (single) fine-grained class of the contrasted objects of S). These communication contexts were created by considering all 707 scenes of ScanNet [63] with all their fine-grained annotated objects classes. Concretely, a context {S, I} of Nr3D satisfies:

1. $2 \leq |o \in S \cap \text{class-of(o)} == I| \leq 6$. In words, the contrasted objects are more than 1 but not more than 6, and they are of the same fine-grained object class.
2. There exist 5 or more scenes, S, for any given I, for which the above condition is satisfied.
3. I is not a structural object class ('wall', 'floor', 'ceiling'), nor a part of an object ('doorframe', 'stair rail', 'closet wall'), nor an object class that tends to have vague or poorly annotated object instances ('object', 'decoration', 'clothes', 'clothing').

D.1.2 Representing ScanNet Scenes on AMT

To create Nr3D, we utilized the web interface presented in Fig D.1. The 3D scene shown to users (acting as speakers or listeners) was a decimated mesh representation of a ScanNet scene. The

high-resolution mesh and low resolution (decimated) mesh are obtained from ScanNet[63] dataset. The decimated mesh is then UV-unwrapped to create texture mapping. The texture is mapped from high-resolution ScanNet mesh to decimated mesh mentioned above and packed into GLTF2 format. This way, we can load the decimated mesh in the browser fast while keeping the high visual quality of the mesh. The users can navigate the scene via rotation, pan, and zoom in any place of the given scene. The rendering is done in real-time through a web browser using WebGL.

D.2 Spatial References in 3D

Sr3D is built on top of ScanNet [63]. In this section, we will discuss the generation method for each spatial relation and how we created human-like utterances out of these relations. In Table D.1, we provide the number of unique communication contexts for each relation type.

Spatial Relation	|Contexts|
closest	17,126
farthest	16,875
between	3,569
front	703
behind	113
left	518
right	546
supporting	390
supported by	357
above	960
under	629
Total	**41,786**

Table D.1: **Detailed statistics of Sr3D.** For each spatial relation we report the total number of unique tuples (communication contexts) it creates in ScanNet.

D.2.1 Horizontal Proximity Based Relations

This type of relation describes what the nearest (closest) and the farthest target objects are according to a unique anchor object in the scene. To generate such relations, we get the list of all anchor objects. Then for each anchor object and each fine-grained target class, we calculate the pairwise distance between each same-class target object and this anchor. If the farthest target object to this anchor object is at a distance of *epsilonGap* greater than of the second farthest target object, this combination

of the anchor and the farthest target object is used as a relation. Similar logic is used to get the closest relations (See Fig. D.2).

D.2.2 Support Relations

The support relations describe whether a target object is supporting (holding) or supported by (held by) an anchor object. To generate these relations for an anchor object, we need first to get the target objects that lie in the vicinity of that anchor top/bottom surface. Then, we find the target objects that their top/bottom surfaces touch the anchor's bottom/top surface, respectively (See Fig. D.3). To check if an object is in the vicinity of the anchor object or not, we first look at the two objects' 2D bounding boxes in the top view, then we calculate the intersection area between them and the ratio of the object's area to the intersection area should be greater than a certain threshold. For an object to be touching an anchor object, the difference in their bottom/top or top/bottom z surfaces is within a small range.

D.2.3 Vertical Proximity Based Relations

These relations represent whether a target object is considered above or below the anchor object without touching each other. The generation method for this type of relation is quite similar to the support relations, but we make sure that the target and the anchor objects do not touch each other (See Fig. D.4.)

D.2.4 Between Relations

The between relations describe the target objects that lie between two anchor objects (see Fig. D.5). To generate such types, we consider all possible pairs of anchor objects. For each anchor pair, we look at the anchors' 2D bounding boxes in the top view (XY axes) and find the convex hull of those 2D bounding boxes. We search for target objects that satisfy the following conditions: (a) they do not intersect with the two anchors; (b) they exist solely inside the convex hull where none of their distractors are found inside; and (c) intersect with each of the two anchors in the z-axis coordinates.

D.2.5 Allocentric Relations

These relations indicate where a target object might exist with respect to the anchor orientation. For Example, the armchair (target object) is at the right of the TV (anchor object). For generating allocentric relations, we need to know: (a) whether the anchor objects have an intrinsic front view

(e.g., armchair) or not (e.g., stool); and (b) the orientation of the objects in ScanNet. For (a), we used the annotations of PartNet to extract if a chair has a back or not so as to define which ShapeNet chair models we will use. Also, we manually annotated several ShapeNet models covering 33 categories in total. For (b), we utilized the Scan2CAD [23] annotations that provide 9DOF alignments between ShapeNet models and ScanNet objects. For every anchor object that has an intrinsic front view, we create four oriented sections (regions) (see Fig. D.6), and we try to find the objects that solely occupy an oriented section where no distractors of the same object class co-exist in there. For an object to occupy an anchor's oriented section solely, the ratio of its points inside the section over its total number of points should be greater than a certain threshold (occupancy threshold).

D.2.6 Converting Spatial Relations to Natural-like Utterances

To create natural (human) -like utterances from the extracted spatial (geometric) relations, we use a manually curated set of template-based sentences. Specifically, for each type of spatial relation we created at least 5 template sentences. Given a target object and spatial relation type/anchor we create 2 utterances by replacing the "target" and "anchor" placeholders in a sampled template with the relation's target and the anchor instance types, respectively.

D.3 Implementation Details

We use 4 graph-convolutional layers for DGCN, each producing an intermediate representation of 128 dimensions. D_L, D_V are also 128-dimensional each. We set the α and β hyper-parameters controlling the contribution of the object and text classification losses in the total loss to 0.5 each. To process the linguistic information in our networks, we use a uni-directional LSTM cell [106] with 128 hidden units and word embeddings of $64D$ that were randomly initialized from unit-normal Gaussian. We note that initializing the embeddings with a $100D$ GloVe [207] embedding pre-trained on the 6B Wikipedia 2014 corpus did not give any significant performance boost. For the object referential loss, we use an MLP([128, 64, 1]) network. We sample 1024 points from the point cloud of each segmented object before passing it to the object encoder.

D.3.1 Preprocessing utterances

We preprocess the collected human utterances by i) lowercasing, ii) tokenizing by splitting off punctuation, iii) replacing tokens that appear less than three times in the training split with a special

symbol marking an unknown token (*UNK*). Furthermore, we ignore all utterances comprised by more than 24 tokens (99th percentile) and those for which the human listener in the underlying trial did not guess the target correctly.

D.3.2 Training details

We use Adam [129] with an initial learning rate of 0.0005 and $\beta_1 = 0.9$ across all our experiments. We train each model for a *maximum* of 100 epochs. We use the test-set to evaluate performance at the end of each training epoch and stop the training if we encounter 10 consecutive training epochs without improvement in terms of test accuracy. Our batch size is 32 for all experiments, except for when we train a model with Nr3D and Sr3D utterances (simultaneously) where we use 64 examples in each batch. Last, we use a learning-rate scheduler that reduces the learning rate by a multiplicative factor of 0.65 every 5 consecutive epochs that the test accuracy does not improve.

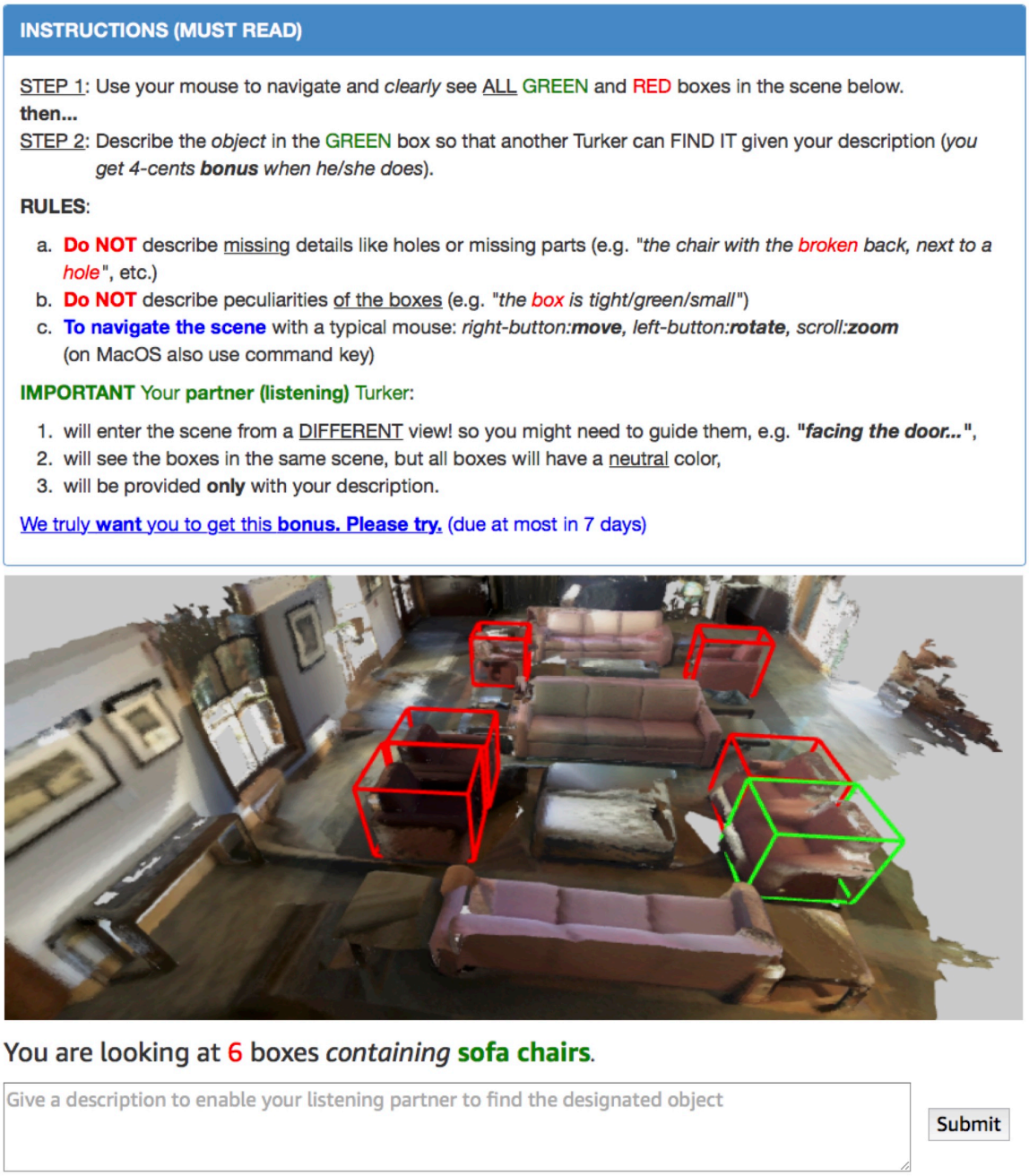


Figure D.1: Snapshot of the interface we used in Amazon Mechanical Turk to collect human utterances while building the **Nr3D**. The Turkers were instructed to pay attention to *all* objects in highlighted boxes while ignoring any sampling artifacts (e.g., holes or broken pieces of the objects). Furthermore, we *motivated the Turkers* to be effective by providing them a financial bonus *(*50% *of their base-pay)* each time their produced utterance enabled the paired listener to guess the target.

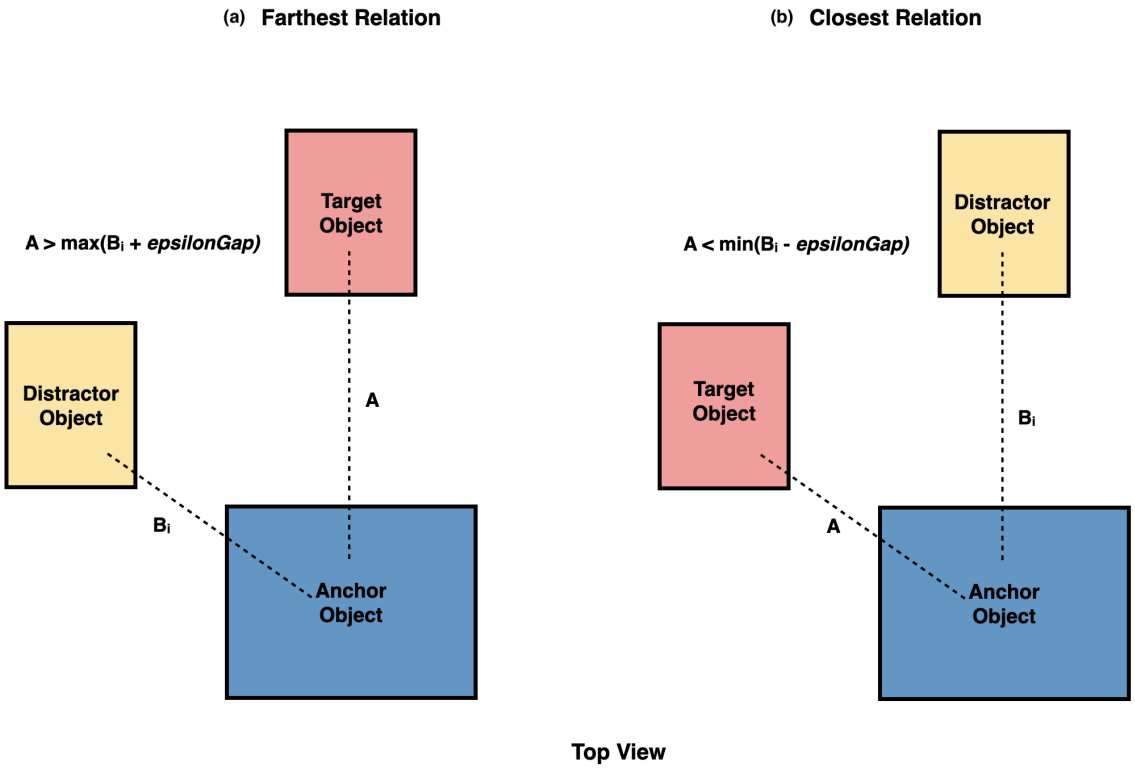


Figure D.2: This figure shows the horizontal (farthest/closest) relations. In (a), the target is the farthest from the anchor object. In (b), it is the opposite. The farthest target to the anchor object should be at a distance greater than *epsilonGap* from the distance between the farthest distractor object and the anchor object.

Figure D.3: Example of support relations. In (a), the target is supported by the anchor object and in (b) the target is supporting/holding the anchor object.

Figure D.4: Examples of vertical (above/below) relations. In (a), the target is above the anchor object and in (b) it is the opposite. The target and the anchor objects should not be touching each other.

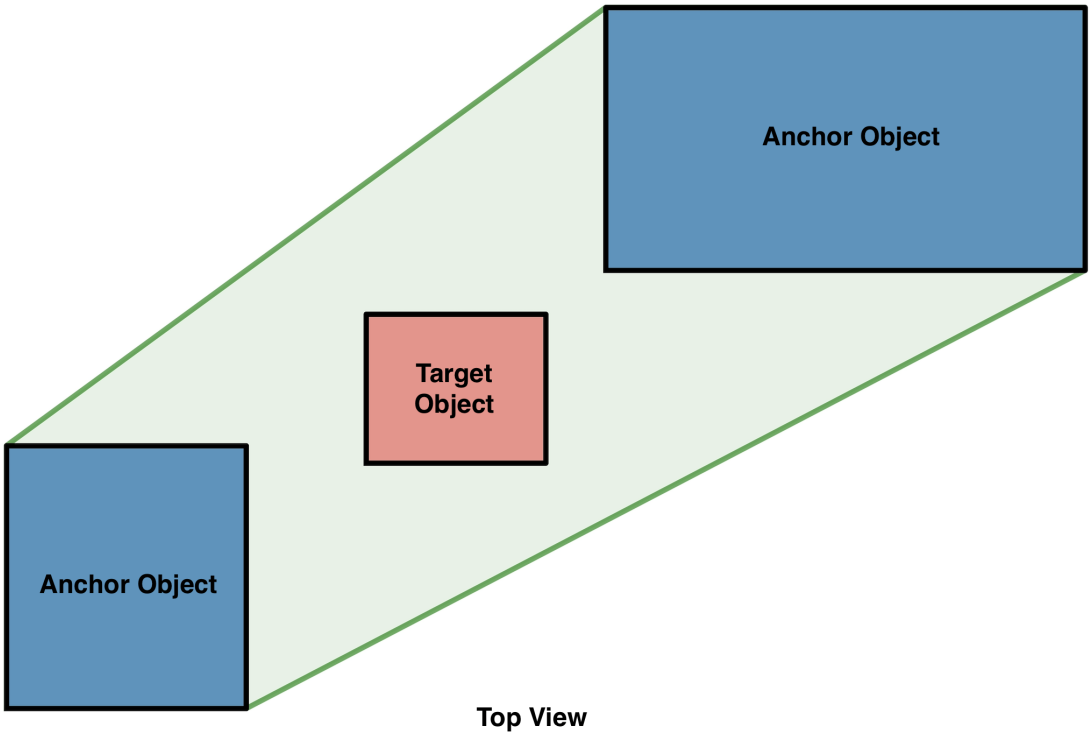


Figure D.5: Example of a between relation. The green shaded area is where a target object should be found to be considered between the two anchors. This shaded area is the convex hull of the two anchor bounding boxes in the top view. None of the target object's distractors should be inside the shaded area.

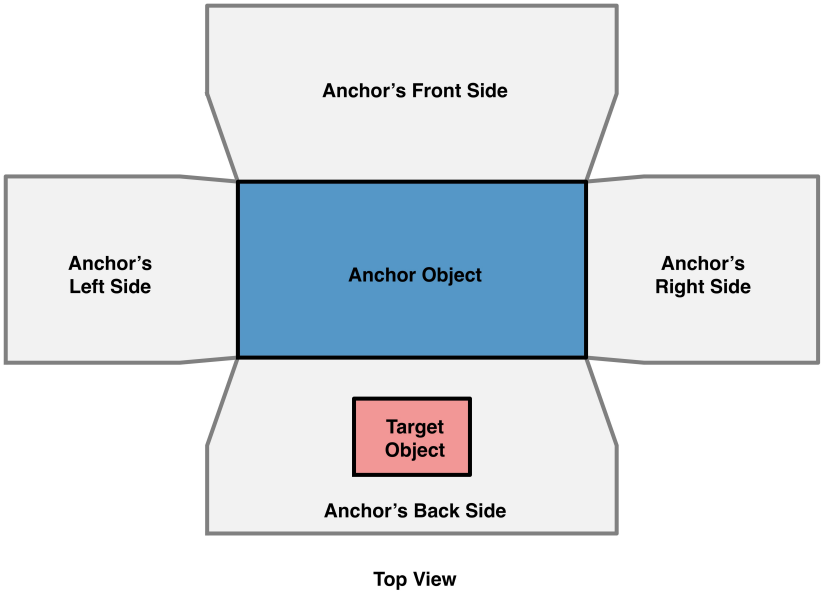


Figure D.6: Allocentric relations generation. This figure shows how we determine where the target object might exist with respect to one of the four oriented sections of the anchor (front, back, left, and right). In this example the target object is at the back of the anchor object.