Graphs, Geometry, and Learning Representations

Navigating the Non-Euclidean Landscape in Computer Vision and Beyond

Geri Skenderi

Supervisor: Prof. Marco Cristani

Department of Computer Science
University of Verona

This dissertation is submitted for the degree of
Doctor of Philosophy

June 2024

Acknowledgements

I have so many people to thank for putting me in the position to write this thesis and it is truly challenging to include all of them on this page. First and foremost, I would like to thank my supervisor, Prof. Marco Cristani for his support during my PhD at the University of Verona. His trust in me and the scientific freedom he allowed were decisive factors in all my work so far. Most importantly, he changed my life by convincing me to pursue a PhD. As of this writing, I cannot imagine myself doing anything else, and for this, I will be grateful forever.

I would also like to express my deepest gratitude to Luigi Capogrosso, with whom I shared many ideas and late nights working in the lab. Along with him, I want to acknowledge all of my labmates during this PhD: Andrea Avogaro, Andrea Toaiari, Christian Joppi, Federico Cunico, Federico Girella, and Francesco Taioli. I was fortunate to have you all as colleagues; your questions and desire to learn pushed me to keep working hard. Most importantly, you were great friends and created an outstanding working environment.

I am very grateful to Prof. Jiliang Tang, who warmly welcomed me into his lab at Michigan State University and showed me new problems and possibilities. Many thanks to all of my friends at the MSU Data Science and Engineering lab, especially Haoyu Han, Harry Shomer, and Jay Revolisnky, for the ideas and discussions we shared.

During this PhD, I was blessed to work with some great people that I have not mentioned so far and whom I would like to acknowledge: Alessio Del Bue, Alessio Sampieri, Guido Maria D'Amely di Melendugno, Fabio Galasso, Francesco Giuliari, Francesco Setti, Franco Fummi, Marco Godi, Matteo Denitto, Simone Melzi, and Yiming Wang. Thank you for letting me be your coauthor and teaching me many things.

A big thank you goes to my closest friends during these past four years: Carlotta Nardin, Emanuel Shushku, Kristi Pina, Luigi Marroccoli, Marco Stridi, and Mattia Badano. Sharing my time and experiences with you is always fun and keeps me in check with reality.

Last but not least, I would like to express my heartfelt gratitude to my parents for their unwavering encouragement and support. Without you, I would have never been able to do anything I have done so far.

I dedicate all of my work to my little brother, Kristian Skenderi. Being your big brother is my favorite thing in the world. Thank you.

Abstract

Artificial Intelligence (AI) requires machines capable of learning and generalizing from data without being explicitly programmed to do so, giving rise to the field of Machine Learning (ML). ML systems follow an inductive process where they learn to make predictions from data, guided by an objective function that defines correct or incorrect choices. This thesis deals with aspects of the subfield of Deep Learning (DL) through Neural Networks (NNs), encompassing the philosophy of emulating the human brain's computational processes.

Most modern NNs excel in solving problems associated with data living in grids with Euclidean properties, such as images, text, and waveforms. However, non-Euclidean data is ubiquitous. A general representative of such data are graphs, i.e., data structures where pairwise relationships between entities, called nodes, are modeled through their connectivity, given by edges. Social networks, molecules, road networks, human-body poses, and 3D point clouds are common examples of data that can be represented through the structure of a graph. To maximize the effectiveness of NNs on data such as graphs, it is imperative to leverage the inherent geometric properties of the given structure. Moreover, geometry can serve as a tool not only to properly understand the input data but also to alter the latent representations space of NNs, giving rise to different desirable properties that can be useful for particular tasks.

This thesis bifurcates into two main paths within the realm of Geometric Deep Learning. The first path explores the application of DL to graph-structured data to solve challenging problems in Computer Vision. The second path delves into the utilization of geometric constraints to shape latent space representations, showcasing how altering latent geometry can give rise to unique and superior solutions in various contexts like Graph Self-Supervised Learning and Multi-Task Learning.

In summary, this thesis navigates the intersection of DL, graphs, and geometry, offering new solutions that enhance the capabilities of NNs in handling non-Euclidean data structures and learning representations that go beyond the commonly assumed Euclidean latent space. Insights from our research reveal both the potential and challenges that lie beyond intuitive geometry and how we can enable ML systems to effectively learn and generalize to a broader range of data types and tasks.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

"Machine intelligence is the last invention that humanity will ever need to make."

Nick Bostrom

1.1 Deep Learning on structured data

1.1.1 Machine and Deep Learning

In the quest for Artificial Intelligence (AI), the ability of a computer program to learn and generalize from previous experience without being explicitly programmed to do so is paramount. Machine Learning (ML) is the scientific discipline that studies this problem. ML systems rely on a computational model mimicking how information manifests in the world, as documented by some data. The data is therefore used to fit the model, i.e., learn a set of parameters that ultimately lead to inducing decision rules, allowing the ML program to make predictions. In practice, this often entails figuring out concentrations in probability mass or density in the joint distribution of the data observations. Machine learning approaches are conventionally categorized into three broad groups, each aligning with a learning paradigm that is based on the available feedback for the learning system:

- Supervised learning: In this scenario, the system receives example inputs and their desired outputs, in terms of (X, Y) pairs, where X is an input random variable and Y is a label that we wish to predict given X. The objective is for the system to discern a general rule that effectively maps inputs to corresponding outputs, and learning is guided directly by how far the predictions deviate from the ground truth Y.

- Unsupervised learning: This approach involves a learning objective that does not rely on labels. The algorithm is left to uncover patterns and structures within its input autonomously, i.e., we only have X at our disposal. Unsupervised learning serves both as an independent goal, to discover hidden patterns in data, and as a means to an end, facilitating feature learning.

- Reinforcement learning: In reinforcement learning, an agent engages with a dynamic environment, striving to achieve a predefined goal (e.g., driving a vehicle or playing a game). Throughout its exploration of the problem space, the program receives feedback akin to rewards, which it seeks to maximize.

While each algorithm possesses distinct advantages and limitations, no single algorithm universally addresses all problems. We will focus on the first two, which are the most popular and well-known approaches. Reinforcement learning, on the other hand, deals with tasks that have to do with exploring and interacting with physical space and is often more helpful in robotics.

In the past, ML practitioners have primarily depended on manually crafting representations, leveraging human insights into the problem. In this thesis, we will be concerned with a particular paradigm of ML known as Deep Learning (DL), which belongs within the broader context of representation learning [21]. DL belongs to a class of learning models that are based on "connectionism", stacking layers of computational units that are interconnected and share information to unravel and interpret hierarchical structures within the input data [135]. The goal behind the driving force of DL models, (Artificial) Neural Networks (NNs), is to try and loosely emulate computation as performed in the human brain, which can be considered the most intelligent connectionist machine in existence [45].

NNs are inspired by synapses and impulse responses found in our neurons. The base computing unit consists of several learnable weights (parameters) associated with every input feature, called a layer. The combined value of the processed information of the layer passes through an activation function that determines the behavior of the output value(s). By stacking several of these layers and using non-linear activation functions, we obtain what is known as a Deep Neural Network. Please note that unless explicitly mentioned, from this point forward we will use NN as an acronym for these Deep Neural Networks. The output of the NN is a prediction of the label Y in the supervised setting or a prediction over X in the unsupervised setting, depending on the objective (clustering, self-supervised learning, etc.). This process is referred to as forward propagation. Given this output and a *scalar* loss function that quantifies how much the output deviates from what we consider ideal, we can train NNs via an optimization process that renders this loss as small as possible. In practice,

this is done through the gradient of the loss function by relying on the well-known Stochastic Gradient Descent algorithm. This process is known as backpropagation [275]. NNs have two properties, among many others, that have made them the most successful models in modern ML:

1. They are universal function approximators [93]. Given some data, a neural network can represent a wide variety of functions or even the data-generating process if it learns appropriate weights. This makes NNs scale exceptionally well with the amount of data;

2. By transforming the input features between the input and output layers, NNs are capable of learning new representations of the data to best solve the task. This means that at each layer, the network learns new features that can facilitate the process of fulfilling the demand of the loss function.

The points presented above are the main reasons behind the modern approach for successfully training these models, which consists of collecting a large amount of data and training an architecture with a large number of weights (parameters). Despite the wide adoption and success they have seen over recent years, NNs still face numerous shortcomings. Of particular interest in this thesis is the fact that by assumption, most architectures are designed to work with data that is represented on a "grid" of numbers. Perfect examples of such data are images, text, and waveforms. This representation facilitates computational operations, and thus, a direct approach is to extend our intuitive, three-dimensional understanding stemming from Euclidean Geometry to the data and representation space. From a computational perspective, this geometry boasts numerous advantages, such as the efficient computation of distances or inner products. At the same time, the linearity inherent in Euclidean space offers a wide array of mathematical tools that can be directly applied without any hassle, such as Linear Algebra and elementary Probability Theory.

Most importantly, reasoning in Euclidean terms closely aligns with our intuition and understanding of structure. Thus, most of the success of NNs has come by assuming that the input data has a Euclidean structure and similar assumptions have also been made regarding the learned representations. For example, images are represented in a two-dimensional grid, which readily allows for the implementation of a convolution operation with a NN [136]. The learned representations are often assumed to live in a vector space such that two vectors with small Euclidean distance represent two (semantically) similar images [137]. In the following subsection, we will intuitively understand why these assumptions are suboptimal for data with explicit structure, like graphs, and why they can fail to capture implicit structure, such as hierarchy and separability.

1.1.2 Learning on structured data

So far, the words "structured data" have appeared often, but there has not yet been a proper characterization. One possible way to formalize this concept, which is also used in this thesis, will be described in the following chapters. As a general idea, we can refer to the concept of structure provided by [16], where the authors describe the necessity of "structured computations" to learn on data that embodies a broad concept of relationships among elements constituting a data instance. Even though one might think that data represented in grids has simplistic structure, that is often not the case. For example, images, text, or sounds are often assumed to live in a lower dimensional manifold embedded in the high dimensional data space [22]. This naturally implies that the data has to contain some structure that we can exploit to represent changes in the high-dimensional data space onto the lower-dimensional manifold. Therefore, it is up to us to define what the structure should be, implicitly in these cases. To provide some intuitive examples, a Convolutional layer processes a local group of pixels in an image [136], assuming independence w.r.t. distant entities, in a translation-invariant fashion, implying that the result of the convolutions should remain relevant across different localities in the image. On the contrary, Recurrent layers are adopted for sequence or text data to favor temporal dependency. More precisely, their inputs and hidden representations, forming the entities, are related through the dependence of one step's hidden state to both the current input and its previous hidden state [209].

Such architectural designs favor what is known as inductive bias, meaning that the learning process is biased into learning a particular function. Architectures such as CNNs or RNNs that are equipped with such biases have led to groundbreaking performance in many different domains. Nevertheless, popular NNs are, by default, not adequately equipped to deal with data whose structure is *explicit*. This is a massive problem because such data are not only widespread in the real world but also provide a way to frame certain problems naturally [212, 241]. A general way to represent structured data is graphs, i.e., data structures where two or more entities, called nodes, are connected together through edges. This representation principle allows much flexibility since the nodes can have discrete or continuous features attached to them, thereby creating a graph signal, the edges can be directed or undirected, and many more [165]. Social networks, molecules, road networks, and human-body poses are very common examples of data that can all be represented through this idea of an abstract structure representing pairwise connectivity, i.e., the graph.

Given their widespread application, Deep Learning on graphs has witnessed a large amount of interest in recent years [258]. From the scientific DL community, the primary reason for this surge in research interest is that if one tries to adapt a vanilla NN to graphs naively, the performance is subpar. The reason for this is that the structure of the input data

is unknown to the base model of computation. For example, consider again the case of graphs. A function that operates on this data structure should be invariant with respect to node ordering because a graph assumes (typically) no order while also understanding that some nodes are connected with each other and not with others. Additionally, phenomena that can easily represented through graphs, such as 3D scene layouts or molecules, might require additional properties, such as equivariance or invariance to rotations and an understanding of hierarchy. Therefore, the main research questions that arise are how we can design architectures with implicit inductive biases to aid learning on structured data and how we can use these architectures and modify their properties in different applicative scenarios. In the following section, we will introduce the general idea behind a blueprint that reveals how we can perform Deep Learning on graphs and structured data by relying on a very powerful mathematical framework.

1.1.3 Geometric Deep Learning

All of the examples provided so far provide subtle hints toward a tool that can be used to formalize this elusive structure and allow for the creation of proper priors to enable high-performance learning on non-Euclidean data. That tool is geometry. This is a somewhat natural choice since, as suggested initially in the main body of work supporting this view [29], geometry seen under the lens of Felix Klein's Erlangen program is the study of objects and functions that remain unchanged under a class of allowable transformations [104]. Sec. 2 shows how geometry can be effectively embedded in NNs to create what is known as the Geometric Deep Learning blueprint. The general idea of this approach is to learn structured data representation through DL by first defining a set of inductive biases that will lead to symmetry or invariance to a chosen property. For example, Graph Neural Networks [211] generalize the notion of convolution to the graph domain by considering adjacent nodes based on the given graph structure instead of the grid surrounding a pixel [125]. This operation is performed so that the signal of each node is updated as a function of its neighbors in a message-passing fashion [76]. Thus, if a permutation invariant aggregation function is applied to the graph signal learned in this manner, altering the node order will not change the output of the function approximated by the neural network.

As mentioned earlier, it is possible to formalize these concepts, and we shall do so later, based on the Geometric Deep Learning blueprint presented in [29]. It is essential to understand that additional geometric properties can be stacked together when designing a NN. Looking at Graph Convolutional Networks again, one can consider the case where the local message passing is performed on features that are invariant to linear transformations. In this case, the approximated function would be invariant to both linear transformations

and node ordering, allowing for a very flexible way of modeling abstract reasoning in space. Furthermore, this property can be extended to heterogeneous graphs with different signals in a straightforward way, enabling powerful multimodal fusion. This is precisely how we propose solving object localization in partial 3D scenes [80, 79], as detailed later in Sec. 3.

Geometric properties can also be extended to the latent representations that a NN learns. This is a very active field of research that has brought about various influential papers in recent years [71, 218], with continued research interests in representation learning [173] and graph representation learning, in particular [164, 196, 279]. In practice, it is frequently the case that latent representations are assumed to live within a Euclidean space. The primary rationale behind this selection is convenience. Euclidean space is a normed vector space with a well-known metric induced by the Euclidean (L2) norm, it has explicit and efficient formulas for the calculation of distance and inner product, and most importantly serves as an intuitive extension of our visually comprehensible 3D space. Nevertheless, it is not the optimal choice of geometry for all cases. It has been shown that non-Euclidean geometries can be highly beneficial in providing additional structure to the latent space so that it better serves specific tasks, such as representing hierarchical and tree-like structures [71]. Finally, in specific scenarios, it is helpful to think about the representation space using a higher-level construct that comes up often in the context of non-Euclidean geometries, i.e., manifolds. Manifolds are, of course, a topological concept rather than a purely geometric one, and they play a central role in representation learning due to the manifold hypothesis [22]. As with any topological space, they can be endowed with a geometry, and if we think of the latent space in this way, we can use various tools from differential geometry to generalize some very useful concepts, such as the Cartesian structure of the Euclidean plane [179, 69]. In this thesis, we will explore two different cases of geometric manipulations of the latent space: one in the case of graph self-supervised learning [223], where hyperbolic constraints are placed in the latent representations, and the other employing a metric space formalization and orthogonality constraints on data manifolds for auxiliary multi-task learning [222].

1.2 Thesis outline and contributions

1.2.1 General outline

As indicated in the final paragraph of the previous section, this thesis comprises two forking paths that focus on the rapidly evolving field of Geometric Deep Learning, the subfield of DL that focuses on developing algorithms and models for data by exploiting or enforcing geometric inductive biases:

1. The first is the use of graphs and DL to solve prominent problems in 3D Computer Vision (CV), a particular domain that is of great applicative interest. The reason for this adaptation, as will be motivated in the following subsections, is that some of these problems can be naturally framed using a graph representation. We will also see that by treating these problems as graph learning problems, we can improve performance, efficiency, and also easily include additional priors.

2. The second is using geometrical constraints on the learned latent representation space. We will show how altering the geometry of this latent space proves beneficial in specific contexts and also how it offers a flexible framework to that can be adapted to seemingly different end goals, such as Multi-Task Learning or Graph Self-Supervised Learning. While all of these topics fall under the umbrella term of Geometric Deep Learning, they are diverse and present various specific challenges.

While the first part focuses on building the optimal graph representation for the problem and the applications, the second forking path focuses more on theoretical aspects of Geometric Deep Learning, dealing with the geometry of latent representations learned from NNs and architectural priors. In summary, this thesis navigates the intersection of DL, graphs, and geometry, offering novel solutions that enhance the capabilities of NNs in handling non-Euclidean data structures and learning representations that go beyond the commonly assumed Euclidean latent space. We will discuss how the ability to learn non-Euclidean data structures, namely graphs, can significantly benefit 3D Computer Vision. Furthermore, we will also show that the geometry of the learned representations of a neural network can significantly affect the learning outcome. This difference can be used creatively, for example, to search for new tasks directly in a learned latent space or to efficiently learn graph-level vector representations with hyperbolic constraints. Insights from our research reveal how much potential and challenges are beyond the realm of intuitive geometry and how we can enable ML systems to effectively learn and generalize to a broader range of data types and tasks. The idea of geometry as a critical aspect of model learning systems will allow for applying these methods to a broader range of tasks in the future and also lead to novel theoretical connections with the scientific disciplines of physics, where geometry and symmetry play a crucial role.

1.2.2 Learning on graphs for 3D Computer Vision

This subsection will broadly explain the CV tasks we propose to tackle by relying on a graph formulation coupled with Graph Neural Networks (GNNs). CV can be defined at a high level as an interdisciplinary field that deals with how computers can be made to gain a

high-level understanding of digital visual information. Several CV problems can be framed using graphs, especially for 3D data. This thesis will explore the following two problems:

Object localization in partially observed 3D scenes. The localization of unobserved objects given a partial observation of a scene is a fundamental task humans often solve in their everyday lives. The problem can be formalized as the inference of the position of an arbitrary object in an unknown area of a scene based only on a partial observation, i.e., only a partial part of the 3D point cloud of a scene is made available. Intuitively, in such scenarios, the most important thing is the ability to reason abstractly over the scene and the objects it contains, without relying much on observations but more so on commonsense. For example, if a human is asked to find the oven, they would most likely look for it in the kitchen, even if they have never observed that particular kitchen or house. To abstractly reason on the observed data, we propose two scene graph representations [80, 79], containing heterogeneous nodes and edges that embed the commonsense knowledge together with the spatial proximity of objects as measured in the partial 3D scan of the scene. The underlying intuition is that commonsense knowledge extracted from an external knowledge base is not specific to any observed visual scene and thus allows for better generalization at the cost of a coarser localization. Our graph representations are first defined by a spatial graph that is fully connected, with nodes representing the known objects and edges representing the proximity. Then, this spatial representation is further expanded by adding and connecting nodes representing concepts through relevant commonsense relationships extracted from a large knowledge graph named ConceptNet[226].

Based on the above problem formulations, we present two different solutions that rely on different levels of geometric constraints. The first is a two-stage solution [80], which imposes two geometrical constraints. First, we predict the pairwise proximity between the target object node, having an unknown position, and each known object node through a Graph Neural Network (GNN), formulating the task as an edge regression problem. As such, the estimated function is invariant to both node ordering and linear transformations, which allows it to generalize easily to scenes with similar content. We then use a *Localization Module* to compute the target's position based on the pairwise distances. The localization module uses a simplex optimization procedure [24] to estimate the position as the intersection of the circular areas defined by all pairwise object distances. This second geometrical constraint also makes our model agnostic to the coordinate system.

We will then show it is possible to obtain even better results with more relaxed constraints regarding the spatial geometry [79], For this purpose, we change the scene graph representation into a directed graph where the proximity edges have relative directional

positions. This means the approximated functions will be invariant to translations but not rotations. The novel scene graph formulation leads to a more straightforward loss calculation and training procedure, which benefits the encoding of both the geometrical information and commonsense attributes, resulting in better target localization. Moreover, we improve the GNN model by proposing the use of a new attention module that adapts Rectified Linear Attention (ReLA). This approach allows the GNN to dynamically sparsify the graph, maintain high expressive power, and remain efficient in terms of training. Through extensive experiments, we demonstrate that our new method achieves a large increase in accuracy with an 8x speed-up in both training and inference, given the end-to-end training. Our proposed approaches provide state-of-the-art results in object localization in partially observed 3D scenes, beating various baselines by a large margin. Finally, given the attention-based model [240] of our best-performing solution, we can get insights into the importance of different parts of the graph signal.

3D Human Pose forecasting. Consider an Industry 4.0 scenario where collaborative robots (cobots) and humans share the same workspace and perform actions concurrently. While there is a clear advantage in increased productivity due to the minimization of idle time, there are also risks of contact and clashes between humans and cobots. Thus, to seamlessly and efficiently interact with human co-workers, cobots need to make decisions on the fly by anticipating the pose trajectories of their human co-workers and predicting future collisions. The 3D human pose can naturally be represented as a graph by encoding the body kinematics, with all joints at all observed frames being the nodes while the edges connect joints in space and time. Therefore, we propose a novel graph-based solution called Separable-Sparse Graph Convolutional Neural Network (SeS-GCN) for human pose forecasting [210]. We design SeS-GCN with performance and efficiency in mind by bringing together three main modeling principles for the first time: depthwise-separable graph convolutions, space-time separable graph adjacency matrices, and sparse graph adjacency matrices. In SeS-GCN, separable stands for limiting the interplay of joints with others (space) at different frames (time) and per channel (depth-wise). For the first time, sparsity is achieved by a teacher-student framework.

The reduced interaction and sparsity result in considerably fewer parameters than a standard GCN (4x less), making our model lightweight, fast, and, most importantly, accurate. To validate the impact of such a model in a real industrial scenario, we also introduce the very first benchmark for human-robot collaboration (HRC), Cobots, and Humans in Industrial COllaboration (CHICO). CHICO includes multi-view videos, 3D poses, and trajectories of the joints of 20 human operators in close collaboration with a KUKA LBR iiwa robotic arm, within a shared workspace. When tested on CHICO, the proposed SeS-GCN outperforms all

baselines with an impressive run time of 2.3ms. Both contributions serve as a stepping stone to cobot awareness in the future, which is instrumental for HRC in industrial applications. Moreover, our proposed solution demonstrates, in line with previous literature [225], that spatiotemporal processing through GNNs is rendered extremely simple and leads to improved representational power for data such as the 3D human pose. Our promising graph-based solution and novel benchmark led to this work being accepted at the European Conference on Computer Vision 2022.

1.2.3 Graph representation learning via hyperbolic self-predictive tasks

In this part of the thesis, we will look at research insights on dealing with both non-Euclidean data and latent representations on graphs in the context of Self-Supervised Learning (SSL). Most NNs on graphs are usually trained by using ground-truth labels. The growing amount of graph data in fields such as bioinformatics, chemoinformatics, and social networks has made manual labeling laborious, sparking significant interest in unsupervised graph representation learning. One particular learning technique in this domain is SSL [261]. In SSL, alternative forms of supervision are created stemming from the input signal [155]. This process is then typically followed by invariance-based or generative-based pretraining. Invariance-based approaches optimize the model to produce comparable embeddings for different views of the input signal. On the other hand, generative-based pretraining methods typically remove or corrupt portions of the input and predict them in data space. Inspired by the recently proposed Joint-Embedding Predictive Architecture (JEPA) [51], we propose Graph-JEPA, the first JEPA for the graph domain, to learn graph-level representations by bridging contrastive and generative models. Joint-Embedding Predictive Architectures are an extremely recent design for SSL. The idea is similar to both generative and contrastive approaches, yet JEPAs are non-generative since they do not directly predict in data space but in the latent representation space. These models can, therefore, be understood as a way to capture abstract dependencies between views of the input signal. However, the graph domain presents us with additional challenges, namely view extraction, designing a latent prediction task that is optimal for graph-level concepts, and learning expressive representations.

In response to these questions, we equip Graph-JEPA with a specific masked modeling objective. The input graph is first divided into several subgraphs, and then the latent representation of randomly chosen target subgraphs is predicted given a context subgraph. The subgraph representations are consequently pooled to create a graph-level representation that can be used for downstream tasks. Thus, Graph-JEPA consists of two encoder networks that receive the context and target subgraphs, respectively and produce the corresponding representations. Notably, these encoders can be different GNNs and don't need to share

weights. A predictor module outputs a prediction of the latent representation of the target given the context. Graph-JEPA does not require any negative samples or data augmentation, and by operating in the latent space, it avoids the pitfalls associated with overfitting in generative models. Given that the nature of graph-level concepts is often assumed to be hierarchical [268, 185], we conjecture that the typical latent reconstruction objective using the distance in a Euclidean space is suboptimal for downstream performance. To this end, we design a prediction objective that starts by expressing the target subgraph encoding as a high-dimensional description of the hyperbolic angle. The predictor module is then tasked with predicting the target's location in the 2D unit hyperbola. This prediction is compared with the target coordinates obtained by using the aforementioned hyperbolic angle. Graph-JEPA outperforms popular contrastive and generative graph-level SSL methods on different datasets while maintaining efficiency and ease of training. Our experiments show that Graph-JEPA can run up to 1.45x faster than recent generative methods. Finally, we empirically demonstrate Graph-JEPA's ability to learn highly expressive graph representations, showing it almost perfectly distinguishes pairs of non-isomorphic graphs that are indistinguishable from common GNNs.

1.2.4 Learning new auxiliary tasks from the representation geometry

Finally, we will explore the connections between latent geometrical constraints and auxiliary learning with NNs. Human learning is often considered to be a combination of processes, such as high-level acquired skills and evolutionary encoded physical perception, that are used together and can be transferred from one problem to another. Inspired by this, Multi-Task Learning (MTL) [36] represents a machine learning paradigm where multiple tasks are learned together to improve the generalization capabilities of a model. A specific form of this learning approach is auxiliary learning [149], which has garnered considerable interest in recent years. In particular, auxiliary learning is a specific type of MTL, where auxiliary tasks are intentionally crafted to boost the performance of a known principal task. As of now, auxiliary tasks are usually either manually defined or found through procedures based on meta-learning [153, 188], a complex learning technique that requires a prior definition of the hierarchy of the auxiliary tasks. Both approaches are pretty inefficient in terms of manual labor and computation, respectively. We tackle this difficult problem by proposing Detaux, a strategy that discovers unrelated auxiliary classification tasks through latent space disentanglement. Given a collection of input data, such as a set of labeled images, a disentanglement procedure should output a representation where the different generative factors that produce the variations observed in the data are separated. Our idea is to work in a specific geometric factorization of the representation space, namely a product manifold,

to unveil auxiliary tasks for the given principal task. The product manifold represents a *disentangled* (factorized) latent space such that each submanifold that composes the product manifold corresponds to specific variations in the data.

One can also consider a finite-dimensional normed vector space and factorize it into vector subspaces (such spaces are indeed a particular case of a manifold). Therefore, using Euclidean subspaces can be considered a special (yet efficient) case of the disentanglement framework. Our method starts by first forcing the variation corresponding to the principal task labels in one submanifold. Then, we automatically identify the most disentangled subspace that is not that of the principal task. Finally, we generate a new auxiliary task and its labels via a clustering module to enable seamless integration with the primary task in any MTL model. The geometry imposed upon the latent space guarantees that the discovered task is unrelated (orthogonal) to the principal task, which has shown to be quite effective in previous literature [249, 282, 197, 114, 281, 151]. Moreover, our proposed approach is agnostic to the choice of the MTL model, given that the latter acts directly on the primary and auxiliary labels. Therefore, the proposed method is fully automatic and offers great flexibility in terms of modeling and design choices, which can depend on several factors such as performance, efficiency, scalability, and resource constraints. Our experimental validation shows that three different MTL models offer improved performance with Detaux and that the disentangled representation is a key factor for data-driven auxiliary task discovery.

The rest of this thesis is organized as follows: the following chapter will provide the necessary background on geometry, show how it connects with the Geometric Deep Learning blueprint, and finally present how this blueprint is realized in the case of Graph Neural Networks. Afterward, the aforementioned research projects will be presented in depth in the same order as presented above: Chapter 3 will deal with our proposed uses of GNNs in CV, chapter 4 will present Graph-JEPA, and Chapter 5 will discuss Detaux. Each chapter will contain any additional background or technicalities necessary in order to make this document as self-contained as possible. Finally, Chapter 6 will delve into an analysis of our research, its possible industrial impact, future work, and a few opinions on the intersection of Graphs, Geometry, and Learning Representations.

Chapter 2

Background

Most of the material presented in this chapter is presented in additional detail by [104] and [29]. The overall goal of this chapter is to present the concept of geometry as a modern mathematical object, initially ideated by Felix Klein in the Erlangen program. By having a deeper understanding of what geometry actually is, we shall explore how this definition can be coupled with DL to create invariant or equivariant NNs with respect to a particular geometry. Finally, we will explore how this way of defining priors for a learning system ties into Graph Neural Networks and their permutation invariance properties. Ideally, the reader should complete the chapter knowing the formalization of a geometry and how to reason in terms of invariances, how these concepts can be used in DL through the Geometric Deep Learning Blueprint, and how Graph Neural Networks can be seen as having a geometric inductive bias. The reader can additionally consult [163] and [5] for supplementary information regarding the characterization of groups and geometries, with a particular focus on hyperbolic geometry.

2.1 Geometry

The fascination of humans with shapes, their properties, and how to construct them is as old as recorded history. The pyramids of Egypt, the aqueducts of Rome, and the Great Wall are just a few examples of how humans used shapes to understand and give form to desired physical attributes. Scientifically speaking, this renders geometry one of the earliest branches of mathematics, along with the creation of numbers and basic algebraic manipulations. Geometry finds its cornerstone in Euclid's Elements, a seminal text [220]. Following the classical (yet unprecedented) Greek mathematical method, Euclid proceeds by using a rigorous logical approach, proving every definition based on his five postulates. Euclid's Elements held its status as *the* definitive geometry text for over two millennia, celebrated for its brilliance in logical reasoning and overall scientific impact. However, one

of Euclid's five postulates, i.e., the parallel postulate, became a focal point of intense debate, eventually paving the way for the development of non-Euclidean geometries. The discovery and establishment of non-Euclidean geometries was due to contributions from many great mathematicians such as Gauss, Bolyai, Lobachevsky, and Poincaré, while the impact of these mathematical constructs arguably reached its peak in science when Albert Einstein used them to establish his theory of relativity.

After non-Euclidean geometry became an accepted and established concept, various mathematicians argued about how geometry was constructed and what it truly meant. The most important contribution in this sphere, which we will also rely on, was from Felix Klein, who approached the subject in a unique way that led to a general and intuitive understanding. While Euclid and most of his successors employed an additive method and approached geometry by establishing fundamental axioms and subsequently constructing a sequence of results based on prior ones, Klein adopted a subtractive approach. Klein started with a space, or more broadly, a set of possible figures, and then defined a set of permissible transformations for that space. He then eliminated concepts that did not retain their identity under these transformations. According to Klein, the essence of geometry lied in examining objects and functions that remain invariant amidst allowable transformations, i.e., *geometry is the study of symmetry*.

We will finally formalize this idea. Before getting there, we first have to be familiar with the concept of a group. A group is an algebraic object and it is defined with regard to an operation. In the context of geometry, it is more valuable and practical to consider a collection of transformations that act on some set.

Definition 2.1.1. *(Group of transformations): A collection G of transformations of a set A, i.e., a collection of bijective functions with A as domain and codomain, is called a group of transformations if it satisfies the following axioms:*

1. Closure: If $T, S \in G$, then we have $T \circ S \in G$.
2. Associativity: If $R, S, T \in G$, then $(R \circ S) \circ T = R \circ (S \circ T)$.
3. Identity: G contains an identity transformation $T : A \to A$ such that $\forall a \in A$, $T(a) = a$
4. Inverse: If $T \in G$, then there exists an element denoted by $T^{-1} \in G$, such that $\forall a \in A$, $T^{-1}(T(a)) = a$.

If the composition of transformations in G is also commutative, i.e., $\forall T, S \in G$, $T \circ S = S \circ T$, the group is called Abelian. As mentioned earlier, readers who are not familiar with group theory might be somewhat puzzled by the above definition, as a group is usually

defined by considering a set A and an operation $\otimes : A \times A \to A$, such that the group is the tuple $(A, \otimes)$. The axioms are then defined similarly. In our case, we can consider each transformation as a valid group operation. Then, the group consisting of all transformations that can be composed together defines G. Our choice of notation is because we wish to consider the group structure on *transformations* of a given space A. There are many reasons why we want to rely upon this algebraic structure, but the main ones are the last two group axioms, which indirectly encapsulate a general idea of symmetry. The high-level intuition is that if we have a group of transformations, we can define elements of A that are unchanged by them and also define "equivalent" structures due to the existence of the inverse axiom (the correct word for this equivalence would be isomorphic). Without further ado, we can now define geometry as formalized by Klein.

Definition 2.1.2. *(Geometry): Let S be any set, and G a group of transformations on S. The pair (S, G) is called a geometry. A figure in the geometry is any subset $A \subseteq S$. An element of S is called a point in the geometry. Two figures A and B are called congruent, denoted $A \cong B$, if there exists $T \in G$ such that $T(A) = B$.*

The above definition shows how Klein's approach starts with a collection of elements (S) and then defines a group of " admissible " transformations for those elements. This definition might seem puzzling at first. How is this abstract idea of the group of transformations related to shapes and what we are used to calling geometry? The best way to illustrate the utility of this definition is through some examples. In the final part of this section, we will explore two concrete realizations of this definition: Euclidean and Hyperbolic geometry. At this point, it is crucial to first state two additional definitions that will be the cornerstones behind geometric priors in DL.

Definition 2.1.3. *(Invariant Set): A collection D of figures in a geometry (S, G) is called an invariant set if, for any figure $A \in D$ and any transformation $T \in G$, $T(A) \in D$. D is called* minimally invariant *if no proper subset of it is also an invariant set.*

The invariant set is an important concept since it formalizes the idea that, given a particular geometry, there exists a collection of figures that always has closure. Intuitively, this collection preserves their structure in the given geometry. Having this collection of figures, we can define the most crucial concept for Geometric Deep Learning:

Definition 2.1.4. *(Invariant and equivariant functions): A function f defined on an invariant set D is called an invariant function if $f(B) = f(T(B))$ for any figure $B \in D$ and any transformation $T \in G$. The function is called equivariant if $f(T(B)) = T(f(B))$.*

To frame the above definition in simple words, the output of an invariant function is unaffected by transformations, while the output of an equivariant function is the same as applying the transformation after the mapping. These two types of function are what we will later try to approximate using NNs, giving rise to the name Geometric Deep Learning. Armed with the above definitions, we can now state a powerful theorem that reveals how we can easily construct minimally invariant sets, which is extremely helpful when reasoning upon the geometric inductive biases we would like to insert into our NN architectures.

Theorem 2.1.5. *An invariant set D of figures in a geometry (S,G) is minimally invariant if and only if any two figures in D are congruent.*

Proof. First, assume D is a minimally invariant set in the geometry (S,G), and let $A,B \in S$ be arbitrary figures in D. We wish to show that any two figures in this set are congruent. Let us start by constructing a new set of figures, the one consisting of A and all transformations of A. In particular, define $\mathscr{A} = \{T(A) \mid T \in G\}$. Notice that for any $T \in G$, $T(A) \in D$ due to D being invariant. This implies that $\mathscr{A} \subseteq D$. Furthermore, due to the group structure of G, $\mathscr{A}$ is also an invariant set. In particular, if $C \in \mathscr{A}$, then $C = T_0(A)$ for some $T_0 \in G$. Thus, we have $T(C) = T(T_0(A)) = T \circ T_0(A)$, for any transformation T that we want to apply to C. Since $T \circ T_0 \in G$ by the group axioms, we have $T \circ T_0(A) \in \mathscr{A}$. Therefore, $\mathscr{A} \subseteq D$ and $\mathscr{A}$ is an invariant set. Since D is minimally invariant, $\mathscr{A} = D$ by Def. 2.1.3. This means that the figure $B \in D$ is also in A, i.e., $A \cong B$, as desired.

The other direction follows a straightforward contradiction. Let D be an invariant set such that any two figures in D are congruent. We wish to show that D is minimally invariant. Suppose that D is not minimally invariant. By Def. 2.1.3, a *proper* subset $D' \subset D$ exists and is minimally invariant. Consider an arbitrary figure A such that $A \in D$ and $A \in D'$. The minimal invariance of D' implies that there exists another arbitrary figure $B \in D$ and $B \notin D'$. If this is the case, given the congruence of any two figures in D, there exists a transformation $T \in G$ such that $B = T(A)$. But this implies that there exists a transformation T such that $T(A) \notin D'$, which in turn implies that D' is not an invariant set. By contradiction, D must be minimally invariant, as desired. □

The theorem presented above presents a convenient way of discovering minimally invariant sets: If A is a figure in (S,G), then $D = \{T(A) \mid T \in G\}$ is a minimally invariant set. Therefore, the transformations of a geometry define congruences and minimally invariant sets, upon which we can learn symmetries of the data. In simpler words, if we consider all the possible and admissible transformations of a figure in a geometry, we can then reason about exciting properties such as invariance and equivariance that we can potentially learn. This is the principal idea behind Geometric Deep Learning, which we will explore in depth

in the following section. To provide the reader with a clear idea of all the material discussed so far, let us look at two of the most popular models of geometry in terms of Def. 2.1.2, given that they will also frequently be mentioned later in this thesis.

Euclidean geometry. Possibly the most well-known and commonly used in practice, Euclidean geometry is the geometry $(\mathbb{C}, E)$, where E consists of all general transformations of the form $T(z) = az + b$, with $a, b \in \mathbb{C}$ being constants and $|a| = 1$ ($|\cdot|$ refers to the modulus of complex numbers). It is quite straightforward to show that this collection of transformations forms a group. Essentially, the group E includes rotations and translations but not dilations. Therefore, a straightforward way to understand Euclidean geometry is thinking of figures and transformations that preserve the Euclidean distance ($\forall z_1, z_2 \in \mathbb{C} : d(z_1, z_2) = |z_1 - z_2|$). This is indeed not the case for dilation, so it is not considered admissible in this geometry.

Hyperbolic geometry. Hyperbolic geometry is one of the most popular models of non-Euclidean geometry. It has many different models; only one will be depicted here to provide a general idea. These different models of hyperbolic geometry are "equivalent" in that they describe the same space in different coordinates, meaning that each can be transformed into the other. Before explaining the Poincaré disk model of hyperbolic geometry, we have to describe Möbius transformations, which are the transformations used to define hyperbolic geometries. Consider the function defined on $\mathbb{C}^+$ (the extended complex plane) by $T(z) = az + b/cz + d$ where a, b, c and d are complex constants. This function is called a Möbius (or fractional linear) transformation if $ad - bc \neq 0$. The complex number $ad - bc$ and is denoted as $Det(T)$. The Poincaré disk model for hyperbolic geometry is the pair $(\mathbb{D}, H)$ where $\mathbb{D}$ consists of all points $z \in \mathbb{C}$ such that $|z| < 1$, and H consists of all Möbius transformations T for which $T(\mathbb{D}) = \mathbb{D}$. The set $\mathbb{D}$ is called the hyperbolic plane. This model describes hyperbolic geometry as the set of all points bounded by the unit disk (i.e., the open unit disk), where the transformation group maps $\mathbb{D}$ onto itself. The defining feature of hyperbolic geometry is that given a point z_0 and a hyperbolic line L not through z_0, two distinct hyperbolic lines exist through z_0 that are parallel to L. This differs from Euclidean geometry's parallel postulate, where only one parallel line can exist.

A great many things can be said about hyperbolic geometry, such as how the transformations are defined through hyperbolic reflections or that hyperbolic geometry is the geometry of negative Gaussian curvature, but that is beyond the scope of this thesis. Geometry in curved spaces is somewhat important for the work discussed in Chap. 5, but more so are the topological spaces where these symmetries operate, which we will detail when the time comes. We refer the reader to [5, 163] for a detailed introduction to hyperbolic geometry.

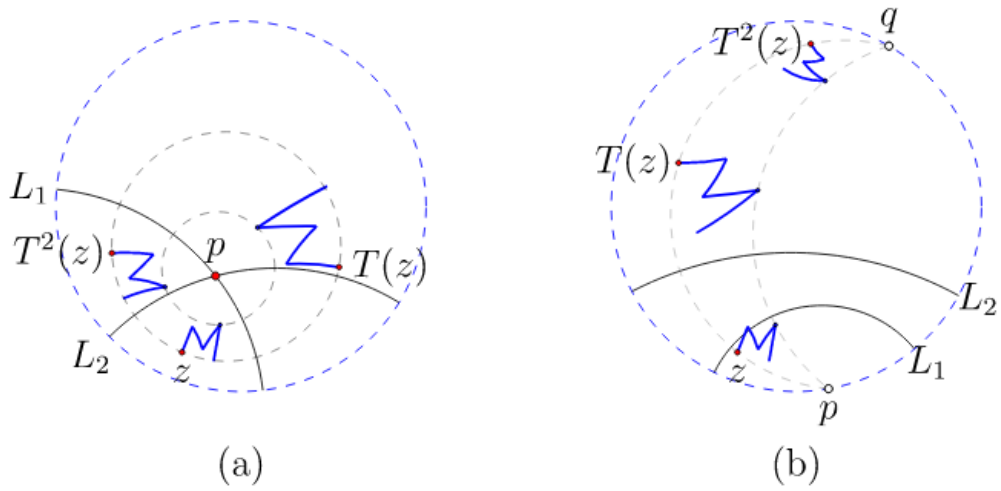


Fig. 2.1 Representative transformations of the letter "M" in the hyperbolic plane, (a) rotation around point p and (b) translation while keeping the ideal points p and q fixed. In (a), the rotation is achieved through two inversions about clines (generalized lines) L_1 and L_2, intersecting at p and forming right angles with the unit circle. The hyperbolic translation in (b) is generated by two inversions about non-intersecting clines L_1 and L_2, meeting the unit circle at right angles. Any point z in $\mathbb{D}$ moves away from p and towards q along the unique cline passing through the three points p, q, and z. Figure taken from [104], full credit is due to the author.

A helpful illustration of hyperbolic transformations is provided in Fig. 2.1 to provide some intuition into the planar Poincaré disk. There are two key features of this geometry that we will use in Chap. 4. The first is that lines in hyperbolic geometry are geodesics, i.e., the shortest path between two points in $(\mathbb{D}, H)$ is along the hyperbolic segment between them. This means that the proximity of the points in this hyperbolic line can serve as a notion of distance in hyperbolic space, similar to how we visualize the Euclidean distance in the Euclidean plane. The second is that hyperbolic space grows exponentially, not polynomially like Euclidean space, rendering it optimal for embedding hierarchical structures [152, 8]. To provide a simple and intuitive explanation, consider how the number of nodes in a binary tree also grows exponentially with its depth. Note that the concepts we have described so far have concentrated on planar geometry but can be generalized to higher dimensions.

2.2 The Geometric Deep Learning Blueprint

In this section, we summarise the construction of geometric priors according to [29], which we will then use to derive, as a particular case, NNs on graphs. These are the most important

learning architectures for this thesis. After going through the rest of the background, the reader will have a better understanding as to how we can embed geometric priors into NNs and why these lead to better performance in various tasks we can obtain better results when applying them to different 3D Computer Vision problems (Chap. 3). Later, we will show that hyperbolic constraints on the representation latent space can work together with the geometric priors in the data space and provide helpful inductive biases for learning on graphs (Chap. 4).

Geometric Deep Learning is fundamentally built upon three principles. Furthermore, as we will detail later, we can extend geometric priors into a NN model by forcing particular structures on the latent space directly. Let us first define our setup in a general way. We wish to approximate a function f acting on some signal through a NN parametrization, such that the approximation f_θ will produce optimal representations (regarding the training objective), where θ denotes the parameters (weights and biases) of the NN. We will follow [29]and adopt the notation $X(\Omega, C)$ to define C-valued signals on the domain Ω. As a simple example, square RGB images can be considered 3-valued signals X defined on $\Omega = \mathbb{Z}_n \times \mathbb{Z}_n$ (i.e., a signal $x : \Omega \to \mathbb{R}^3$). Furthermore, we will assume we prior have knowledge of a symmetry group G that acts on the domain. The main goal behind Geometric Deep Learning is to provide a way of designing NNs that approximate f while satisfying the geometric priors provided by G, meaning that such architectures have strong inductive biases. By knowing the symmetry group G, we can use the concepts introduced in Theorem 2.1.5 and Def. 2.1.4 to understand how the function class we wish to learn must be restricted.

The main principles behind Geometric Deep Learning are symmetry, geometric stability, and scale separation. As mentioned before, symmetry is the most important aspect of the study of geometry. In our setup, it refers to the ability of the learning architecture to understand invariants in the data. Intuitively, this implies that the approximated function f_θ should respect and be aware of the symmetries that are present in the domain, as specified by G. The main idea behind geometric stability, on the other hand, is to express global symmetries in an approximated way through local symmetries, such that f_θ is stable to small deformations in the signals and is thus able to understand symmetries on a smaller scale. Therefore, we can intuitively understand geometric stability as the ability of the NN to operate at a local scale. An excellent example of why this is necessary can be seen in videos, where several objects might be moving in a different direction. In subsequent frames, the resulting scene will contain approximately the same semantic information, yet no global translation can be used to explain the change from one frame to another. This property is particularly important for learning tasks where the domain Ω can change, as is the case for graphs. Finally, scale separation refers to the ability to preserve essential characteristics of

the signal when transferring it onto a coarser version of the domain. The intuition behind this principle is related to the current representation learning process of neural networks. Given that we often assume the learned representations to be dense vectors in a vector space, we would like these dense vectors to maintain the information that was initially processed locally. A simple way of interpreting the previous statement is that pooling-like operations do not cause issues like representation collapse. By combining these concepts, it is possible to describe a general blueprint for learning stable representations of high-dimensional data. The general idea behind the blueprint is as follows: First, we learn representations locally in a way that respects the symmetries in the data. Then, by relying on appropriate pooling operations, we can represent the whole domain with a single vector representation, or skip this step to predict at the local level. At this point, we are ready to dive into a formal description of the Geometric Deep Learning blueprint.

The first building block will be defining a family of linear equivariant functions, which enable the construction of rich and stable features by composition with appropriate non-linear maps. Linearity here is specified with regards to group G, i.e., if we have a transformation $T \in G$, then $T(\alpha x + \beta x') = \alpha(T(x)) + \beta(T(x'))$ for any scalars $\alpha, \beta \in \mathbb{R}$ and signals $x, x' \in X(\Omega, C)$. Then, we can state the following proposition:

Proposition 2.2.1. *If $B : X(\Omega, C) \to X(\Omega, C')$ is a G-equivariant layer (according to Def. 2.1.4) and $\sigma : C' \to C''$ is an arbitrary (non-linear) elementwise map, such that $(\sigma(x))(u) = \sigma(x(u))$, then the composition $U = (\sigma \circ B) : X(\Omega, C) \to X(\Omega, C'')$ is also G-equivariant.*

Proof. This statement can be verified using the definition of the definition of (left) group action. If we have a group G acting on Ω, we automatically obtain a (left) action of G on the space $X(\Omega)$, defined as $(g.x)(u) = x(g^{-1}u), u \in \Omega$ and $g \in G$. Supposing the group action is already defined to act on x, from the group action definition, we get that $g.U(x) = U(g^{-1}x) = \sigma(B(g^{-1}x))$. On the other hand, given the equivariance of B, we can write $U(g.x) = \sigma(B(g.x)) = \sigma(g.B(x)) = \sigma(B(g^{-1}x))$. Therefore, $g.U(x) = U(g.x)$, i.e., U is G-equivariant, as desired. □

This simple property allows us to define a very general family of G-invariants, by composing U with a mathematical construct known as the group average. A natural question that arises is whether any G-invariant function can be approximated at arbitrary precision by such a model, for appropriate choices of B and σ. While this is beyond the scope and purpose of this thesis, results from the literature have shown that it is possible to adapt the standard Universal Approximation Theorems from unstructured vector inputs to show that shallow "geometric" networks are also universal approximators, by properly generalizing the

group average to a general non-linear invariant. Such proofs have been demonstrated, for example, for the Deep Sets model [272].

We are now ready to satisfy the property of geometric deformation stability. For this purpose, we will consider an alternative representation, which considers localized equivariant maps. Assuming that Ω is further equipped with a distance metric d, the equivariant map U defined above is localized if $(Ux)(u)$ depends only on the values of $x(v)$ for $\mathscr{N}_u = \{v \mid d(u,v) \leq r\}$, for some small radius r. It is worth noting that meaningful metrics can be defined on grids, graphs, manifolds, and even groups. The set $\mathscr{N}_u$ is called the receptive field, a term that was rendered very famous by CNNs. A single layer of a local equivariant map U cannot approximate functions with long-range interactions. However, a composition of several local equivariant maps $U_i \circ U_{i-1} \circ \ldots \circ U_1$ increases the receptive field. The receptive field is further increased by interleaving downsampling operators that coarsen the domain (again assuming a metric structure). This renders the NN able to process information locally and be stable to deformations in the signal or domain if the receptive field is defined correctly. This final aspect of coarsening is what also ensures us of scale separation.

In summary, with knowledge of the geometry of the input domain and the underlying symmetry group, we need three fundamental building blocks for optimal geometric priors: (i) a local equivariant map, (ii) a global invariant map, and (iii) a coarsening operator. These building blocks are able to construct powerful NNs with prescribed invariance and stability properties by a combination that the authors in [29] refer to as the Geometric Deep Learning Blueprint. Let Ω and Ω' be domains, G a symmetry group over Ω, and let $\Omega' \subseteq \Omega$ denote the fact that Ω' can be considered a compact version of Ω. The blueprint consists of the following design principles:

1. Linear G-equivariant layer $B : X(\Omega, C) \to X(\Omega', C')$ satisfying $B(g.x) = g.B(x)$ for all $g \in G$ and $x \in X(\Omega, C)$.
2. Nonlinearity $\sigma : C' \to C'$ applied element-wise as $(\sigma(x))(u) = \sigma(x(u))$.
3. Local pooling layer (coarsening) $P : X(\Omega, C) \to X(\Omega', C)$, such that $\Omega' \subseteq \Omega$
4. G-invariant layer (global pooling) $A : X(\Omega, C) \to Y$ satisfying $A(g.x) = A(x)$ for all $g \in G$ and $x \in X(\Omega, C)$

Using these blocks allows constructing G-invariant functions $f : X(\Omega, C) \to Y$ of the form $f = A \circ \sigma_J \circ B_J \circ P_{J-1} \circ \ldots \circ P_1 \circ \sigma_1 \circ B_1$ where the blocks are selected such that the output space of each block matches the input space of the next one. Different blocks may exploit different choices of symmetry groups G. The function composition defined above allows us to (informally) read the Geometric Deep Learning Blueprint as the following

process: "First, operate the local level in a given domain by considering a neighborhood of signals as specified by a metric d. Then apply an element-wise nonlinearity and optionally coarse the domain. Finally, summarize all the signals by using an invariant global pooling operation".

We will now see how this blueprint enables a powerful way of learning over graphs, this thesis's most critical data domain. A higher level description of why the particular form of Graph Convolutional Networks satisfies the symmetries in the graph domain will also be presented by simply looking at the model equations and understanding their equivariance and invariance properties. To conclude this section, it is for the reader to understand that the geometric properties of many popular neural networks that operate on grids, such as the famous translational invariance of CNNs, can be explained using the Geometric Deep Learning Blueprint.

2.3 Graph Neural Networks

At this point, it is finally time to provide a formal definition for graph-structured data. A graph $\mathbb{G}$ can be defined as $\mathbb{G} = (V,E)$ where $V = \{v_1 \dots v_N\}$ is the set of nodes, with a cardinality $|V| = N$, and $E = \{e_1 \dots e_M\}$ is the set of edges, with a cardinality $|E| = M$. It is common for graphs to have node features, i.e., attributes associated with each node $X \in \mathbb{R}^{N \times d}$. The set of edges can also have features, i.e., each connection between nodes is associated with a feature vector that provides information about the connection $\mathscr{E} \in \mathbb{R}^{M \times d}$. What makes graphs special is their explicit structure given by the connection of nodes through edges. This structure is usually represented through an adjacency matrix $A \in \mathbb{R}^{N \times N}$, where $A_{ij} \neq 0$ indicates that nodes v_i and v_j are connected with a weight A_{ij}, $A_{ij} = 0$ otherwise. In the case of unweighted connections, we discretize the adjacency matrix such that $A \in \{0,1\}^{N \times N}$. We will focus on this latter case, but it is straightforward to generalize the operations to weighted graphs.

Graphs are prevalent in various scientific disciplines, ranging from sociology to chemistry, given their ability to serve as models for systems of relations and interactions. They are therefore very important in the field of machine learning, considering this wide range of applications. Given the structure provided by A, graphs inherently embody a fundamental invariance characterized by the group of permutations. This simply means that what we learn on a graph structure should be invariant to the ordering of the nodes, since the order does not alter the structure of the graph. In terms of what we have discussed so far, we would like to approximate functions on a graph such that they are invariant to node permutations, i.e., the function approximator (neural network) understands this geometry of the data.

In summary, for two isomorphic graphs[1], the outcomes of these functions are identical. Functions such as these yield a global or graph-level output. In practice, however, it is frequently the case that we seek to operate on a node-by-node basis. For instance, we might seek to apply a function to update the features of each node, such that we get back a latent vector representation of the nodes. Fortunately, the Geometric Deep Learning Blueprint provides us with a direct way of tackling both scenarios. To operate at the node (local) level, we skip the global pooling step, otherwise, if we wish to operate at the graph (global) level, we utilize it.

Formally, let us consider $\Omega = \mathbb{G}$ as the domain and $X(\mathbb{G}, \mathbb{R}^d)$ as the d-dimensional node signals. The symmetry we wish to consider is given by the permutation group $G = \Sigma_n$, whose elements are all the possible orderings of the set of node indices $\{1, \ldots, n\}$. Under this setting, and given the relationship between A and X (applying a permutation matrix P to the node features X automatically implies applying it to A's rows and columns), we can state that the form of the invariant (f) and equivariant (F) functions we are looking for is:

$$f(PX, PAP^T) = f(X, A) \tag{2.1}$$

$$F(PX, PAP^T) = PF(X, A) \tag{2.2}$$

The Geometric Deep Learning Blueprint supports functions that exhibit locality, meaning that the output for a node u should be directly influenced by its neighboring nodes in the graph. We can explicitly formalize this constraint in the model construction by updating the information of each node based on its immediate neighbors. The 1-hop neighborhood of a node u is defined as $\mathcal{N}_u = \{v | (u,v) \in E \; or \; (v,u) \in E\}$. The neighborhood features can then be defined as $X_{\mathcal{N}_u} = \{\{x_v | v \in \mathcal{N}_u\}\}$. The double parentheses indicate that X is a multiset since it is possible to have identical node features. Operating on 1-hop neighborhoods aligns perfectly with the locality aspect by defining the receptive field over graphs using the shortest path distance between nodes based on edges in E. The recipe stemming from the blueprint for constructing permutation equivariant functions on graphs is to specify a local function ϕ that operates over the features of a node and its neighborhood, $\phi(x_u, X_{\mathcal{N}_u})$. Then, a permutation equivariant function F can be constructed by applying ϕ to every node's neighborhood in isolation:

$$F(X, A) = \begin{pmatrix} \phi(x_1, X_{\mathcal{N}_1}) \\ \vdots \\ \phi(x_n, X_{\mathcal{N}_n}) \end{pmatrix}, \forall x \in X \tag{2.3}$$

[1] In the context of graphs, we can understand the isomorphic property in simple terms as it being satisfied when two different graphs have the same number of nodes, edges, and same edge connectivity

Naturally, the permutation equivariance of F depends on ϕ's output being independent of the ordering of the nodes in $\mathcal{N}_u$, because we don't have a canonical ordering of the neighbors on a graph. Thus, if ϕ is permutation invariant, this property is satisfied. In practice, this requirement can be easily satisfied by defining a neural network that operates at the node level and an aggregation function $\bigoplus$ that is permutation invariant and acts on each neighborhood as defined in Eq. 2.3. At this point, the construction of f becomes trivial. All we need to obtain a graph-level representation is to use a permutation invariant global pooling operator α (e.g., mean, sum, min, max) and define $f = \alpha \circ F$ in the previous section and obtain a unique vector representing $\mathbb{G}$.

Readers familiar with graph theory will notice that this construction looks similar to some very well-known algorithms, such as Label Propagation or the Weisfeiler-Lehman test. Both of these algorithms are message-passing algorithms, which is precisely what Graph Neural Networks are! A concrete view of Graph Neural Networks as message-passing machines is given by the following equation:

$$x_i^{(k)} = \theta^{(k)} \left(x_i^{(k-1)}, \bigoplus_{j \in \mathcal{N}_i} \phi^{(k)} \left(x_i^{(k-1)}, x_j^{(k-1)}, e_{ji} \right) \right), \tag{2.4}$$

where $x_i^{(k)}$ denotes node features of node x_i in layer k, e_{ji} denotes (optional) edge features from node j to node i, $\bigoplus$ denotes a differentiable permutation invariant function, (e.g., sum, mean or max), and ϕ, θ denote differentiable functions such as MLPs. It is clear to see that the composition of ϕ with $\bigoplus$ in the above equation leads us to $F(X,A)$. We will conclude the background by showing how this operation can be understood as a generalized convolution in irregular domains, and prove under this formulation that it can lead to equivariant and invariant functions learned on graphs. The message passing operation above can be expressed in matrix form as the Graph Convolution operation by:

$$\mathbf{X}^k = AX^{k-1}\Theta, \tag{2.5}$$

where Θ represents the weights of the neural network(s). This is of course, a simplified and general representation, which can be modified in different ways. Common examples include learnable adjacency matrices; we would denote the adjacency matrix by A^k. Another famous alternative is the Graph Convolutional Network[125], which modifies the shift operator for additional numerical stability by performing a symmetric normalization of the adjacency matrix via $\tilde{A} = \hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}$, where $\hat{A} = A + I$ denotes the adjacency matrix with added self-loops, and $\hat{D}_{ii} = \sum_{j=0} \hat{A}_{ij}$ the diagonal degree matrix. Note that any elementwise nonlinearity

is also excluded, as we have already shown that it does not affect the equivariance property in Prop. 2.2.1. It is simple to show that this function is permutation equivariant, and can be rendered permutation invariant via composition with a global pooling operation.

Proposition 2.3.1. *A Graph Convolution layer as defined in Eq. 2.5 is permutation equivariant.*

Proof. Let us write the convolution operation as $F_\theta(X,A) = AX\Theta$, where the particular X depends on the layer number. Consider a permutation $P \in \Sigma_n$ being applied to the signal. Then $F_\theta(PX, PAP^T) = PAP^T PX\Theta = PAX\Theta = P(AX\Theta) = PF_\theta(X,A)$, as desired, due to the orthogonality of permutation matrices and the associativity of matrix multiplication. □

Given the above proposition, the following is a straightforward conclusion:

Proposition 2.3.2. *The composition of a Graph Convolution layer as defined in Eq. 2.5 with a permutation invariant (aggregation) function α is permutation invariant.*

Proof. To remain in line with the material presented so far, let us define the new operation as $f_\theta(X,A) = \alpha \circ F_\theta(X,A)$. Consider applying a permutation $P \in \Sigma_n$ to the signal. Then $f_\theta(PX, PAP^T) = \alpha \circ F_\theta(PX, PAP^T) = \alpha \circ PF_\theta(X,A) = \alpha \circ F_\theta(X,A) = f_\theta(X,A)$, as desired, due to the equivariance of $F_\theta(X,A)$ and the assumed invariance of α. □

Armed with this knowledge, it is possible to design powerful representation learning algorithms that respect any assumed symmetries of the graph data domain. In the rest of this thesis, we will see how we can use Graph Neural Networks for node, edge, and graph-level predictions, showing they improve task performance and asserting the importance of geometric inductive biases in Computer Vision and Deep Learning. The following chapter will show how this construction we have derived and additional geometric concepts can be powerful tools to solve complex 3D Computer Vision problems.

Chapter 3

Learning on graphs for 3D Computer Vision

3.1 Object Localization in Partial Scenes

3.1.1 Introduction

The localization of unobserved objects given a partial observation of a scene is a fundamental task that humans solve often in their everyday lives as shown in Fig. 3.10. Such a task is useful for many automation applications, including domotics for assisting visually impaired humans in finding everyday items [62], visual search for embodied agents [15], and layout proposal for interior design [162]. Given the importance of the problem, this chapter will detail how our contributions [80, 79] formally study object localization from a CV standpoint. We formalize the problem as the inference of the position of an arbitrary object in an unknown area of a scene based only on a partial (3D) observation of the scene.

Humans perform this localization task not only by using the partially observed environment but also by relying on the *commonsense* knowledge that is acquired during their lifetime. For example, by knowing that pillows are often close to beds (a *common spatial* relationship) and that chairs and beds are often used for resting (a *common affordance* relationship), one could infer the whereabouts of pillows even if only a bed and a chair were observed. We, therefore, venture into the same direction and ask whether it is possible to computationally solve this task by injecting the commonsense knowledge within a scene graph representation [129, 74, 247], so that a machine can also reasonably localize an object in the unseen part of the scene, without reasoning explicitly on any visual or depth information.

In order to emulate this inference process, we propose a new scene graph representation, the Spatial Commonsense Graph (SCG), having heterogeneous nodes and edges that embed

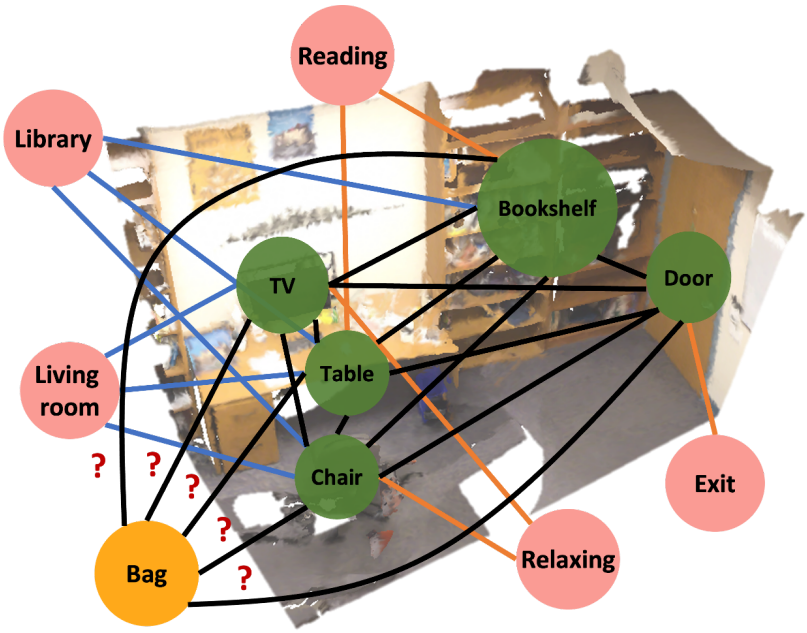


Fig. 3.1 Given a set of objects (indicated in the **green** disks) in a partially known scene, we aim at estimating the position of a target object (indicated in the **orange** disk). We treat this localization problem as an edge prediction problem by constructing a novel scene graph representation, the Spatial Commonsense Graph (SCG), that contains both the spatial knowledge extracted from the reconstructed scene, i.e., the proximity (**black** edges) and the commonsense knowledge represented by a set of relevant concepts (indicated in the **pink** disks) connected by relationships, e.g. *UsedFor* (**orange** edges) and *AtLocation* (**blue** edges).

the commonsense knowledge together with the spatial proximity of objects as measured in the partial 3D scan of the scene. The underlying intuition is that commonsense knowledge extracted from an external knowledge base is not specific to any observed visual scene and thus allows for better generalization, but at the cost of a coarser localization. At the same time, the objects' arrangement in the known portion of the scene is helpful in providing better pairwise object distances, strengthening the estimate of the target object's position. The main challenge here is devising a model that promotes generalization via commonsense and a geometric understanding of the space while increasing the accuracy of the scene-specific metrics.

The proposed scene graph, as shown in Fig. 3.2(a), is first defined by nodes representing the known objects in the scene that are fully connected through edges representing the *proximity*, *i.e.* the spatial relationship between a pair of objects. We call this spatial representation the Spatial Graph (SG) of the known partial 3D scan. Then, the SG is further expanded into the SCG by adding and connecting nodes that represent concepts through relevant commonsense relationships extracted from ConceptNet [226].

The SCG is instrumental in addressing the localization problem. To obtain strong inductive biases, we first propose a two-stage solution, dubbed SCG Object Localiser (SCG-OL). First, the model predicts the pairwise proximity between the target object node, having an unknown position, and each known object node through our graph-based *Proximity Prediction Network* (PPN) by formulating the task as an edge regression problem. We then use our *Localization Module* to compute the position of the target based on the pairwise distances (proximity). The localization module estimates the most probable position as the intersection of the circular areas defined by all pairwise object distances. Note that by only using distances between pairs of objects, our model does not depend on the scene's reference frame, thus being considered agnostic to the coordinate system. Thus, our method is invariant with respect to ordering the scene objects and commonsense entities (Permutation group) and linear transformations of the scene (Euclidean group).

As an extension of the Spatial Commonsense Graph, we also propose a directed formulation dubbed D-SCG, where the proximity edges are represented with directional relative positions (Fig. 3.2(b)). Thus, the model maintains only translational invariance. This difference in the graph formulation allows us to regress and estimate the target position in an *end-to-end* manner, which is not possible with SCG. Intuitively, in this way, we are able to stand in a middle ground between data-driven approximation and strong geometric priors. The novel scene graph formulation also leads to a more straightforward loss calculation and training procedure, which benefits the encoding of both the geometrical information regarding the arrangement of the objects in the observed part of the room and commonsense attributes that define what they are *commonly* used for or where they are *commonly* located, resulting in a better target localization both in 2D and 3D. Given that this extension is more data-dependent, we propose to employ a new attention module in the GNN that is adapted from the Rectified Linear Attention (ReLA) [276] for its high expressive power that encourages the sparsity of attention weights, while being stable and efficient in terms of training. With extensive experiments, we demonstrate that D-SCG Object Localiser improves performance and achieves an increase of with an 8x speed-up in both training and inference compared to SCG-OL. Furthermore, this solution is able to generalize better to the 3D domain, largely due to the end-to-end training procedure.

In order to empirically validate our proposed approaches, we introduce a new dataset built from partial reconstructions of real-world indoor scenes using RGB-D sequences from ScanNet [48]. The dataset is constructed in order to reflect different completeness levels of the reconstructed scenes. We define the evaluation protocol, focusing primarily on the Localisation Success Rate (LSR), a performance measure that quantifies the localization accuracy. To summarise, our core contributions are the following:

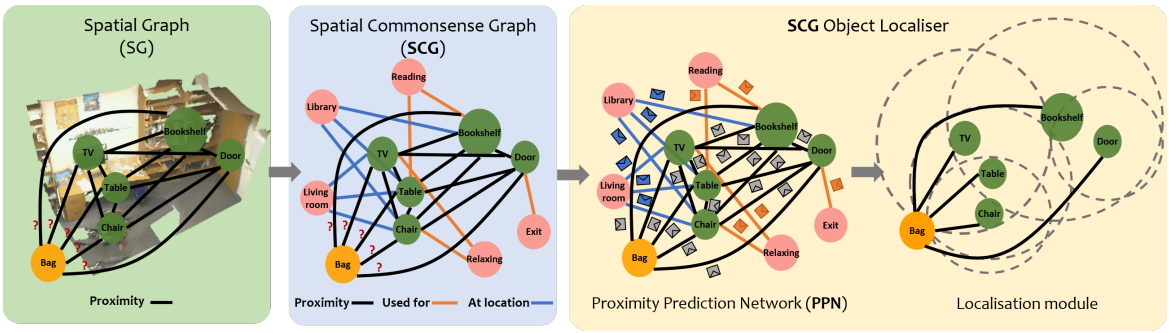
- We identify the novel task of object localization in partial scenes and propose two graph-based solutions. We make a new dataset and evaluation protocol available and show that our methods achieve the best performance compared to other approaches in the literature.

- We propose a new heterogeneous scene graph, the Spatial Commonsense Graph (SCG), for effective integration between the commonsense knowledge and the spatial scene, using attention-based message passing for the graph updates to prioritize the assimilation of knowledge relevant to the task. We also extend SCG by proposing a directed graph formulation, D-SCG, which allows for end-to-end training.

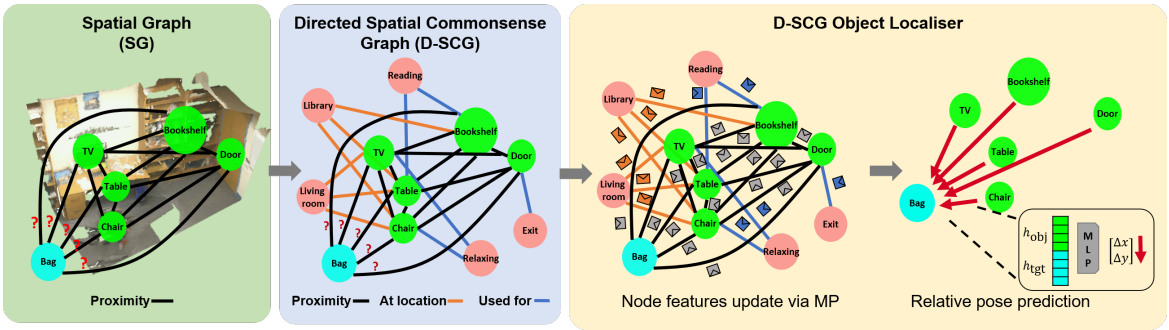
- We propose two different localization approaches that makes use of the graph-based problem formulations. The first, SCG Object Localiser, is a two-staged localization solution agnostic to scene coordinates. The distances between the unseen object and all known objects are first estimated and then used for the localization implemented via circular intersections. This estimation procedure provides robust geometric priors in terms of reasoning over the objects (permutation invariance of the graph neural network) and in terms of spatial awareness (invariance to linear transformations). Based on D-SCG, we also propose D-SCG Object Localiser, where we show that it is possible to construct a data-driven, end-to-end solution. Finally, we propose a sparse graph attention mechanism for D-SCG-OL, which proves beneficial in terms of efficiency and performance.

3.1.2 Related work

Scene graph modeling and inference. Scene graphs were initially used to describe images of scenes based on the elements they contained and how they were connected. The work of [117] showed that for certain applications, e.g. Image Retrieval, the abstraction of higher-level image concepts improved the results compared to using the standard pixel space. Since then, scene graphs have been successfully used in many other tasks such as image captioning [264, 265, 83] and visual question answering [215, 139]. Recently, the use of scene graphs has also been extended to the 3D domain, providing an efficient solution for 3D scene description. The 3D scene graph can vary from a simple representation of a scene and its content, in which the objects are nodes, and the spatial relationships between objects are the graph's edges [74, 247, 257]; to a more complex hierarchical structure that describes the scene at different levels: from the image level with description about the scene from only a certain point of view, moving up to a higher level description of objects, rooms and finally



(a) The Spatial Commonsense Graph (SCG) is constructed from the known scene by enriching the Scene Graph (SG) with concept relationships, resulting in edges of three types: *UsedFor* (**orange** edges), *AtLocation* (**blue** edges) and Proximity (**black** edges). The Proximity edges represent pairwise distances between objects. The SCG is then fed into the Proximity Prediction Network (PPN), which performs message passing with attention to update the node features while taking commonsense into consideration. The PPN then concatenates the node features of the target node and one of the scene object nodes and passes it through an MLP to predict the pairwise distance. The localization module then uses the predicted pairwise distances to estimate the target object's position within the area where most distances overlap.



(b) The Directed Spatial Commonsense Graph (D-SCG) is constructed in an identical manner, with the key difference that the Proximity edges represent pairwise displacement vectors (i.e., the difference in position). The D-SCG is then fed into the D-SCG Object Localiser that first performs message passing with a sparse attention module to update the node features. These features are then concatenated to the node features of the target node and one of the scene object nodes (at a time) and passed through an MLP to predict the relative position. The final position is then given by the average of all the predicted pairwise relative positions (mean pooling).

Fig. 3.2 Complete overview of our two proposed solutions, with their corresponding graph formalization the localization architecture: SCG(a) and D-SCG(b).

buildings [6]. The work of [284] uses a scene graph to augment 3D indoor scenes with new objects matching their surroundings using a message passing approach. A relatively similar task is indoor scene synthesis [251], in which the goal is to generate a new scene layout using a relation graph encoding objects as nodes and spatial/semantic relationships between objects as edges. A graph convolutional generative model synthesizes novel relation graphs and, thus, new layouts. In [55, 162] the authors use a 3D scene graph to describe the object

arrangement, then modify the scene graph and generate a new scene. Like these works, we use an underlying scene representation, but unlike them, we embed commonsense knowledge into the scene graph. This way, our approach can better generalize to unseen rooms with unseen object arrangements by leveraging prior semantic knowledge.

Datasets for Object Localisation. Datasets existing in the literature related to scene graphs are not suited for this type of object localization task. For instance, Scene Synthesis datasets [252] do not have enough variability in the scene structure, as all environments represented are of identical shape and similar size. Moreover, the scenes mostly contain the same set of objects. These characteristics lead to datasets that do not reflect the real world and cannot be used to train models to be deployed in real indoor environments. Another major limitation of existing datasets is their assumption that the entire layout of the room is known and that the objects lie within the boundaries of the observed part of the scene [251, 146], which is atypical. In robotic applications like Visual Search [255, 78, 38], the robot only has partial information about the environment that gets updated during navigation. In general, the searched object has to be found in the unexplored part of the scene, yet to be discovered. Our work is based on partially observed scenes and performs localization without navigation.

Commonsense Knowledge in Neural Networks. Commonsense reasoning focuses on imitating the high-level reasoning employed by humans when solving tasks. Typically, we do not only use the information directly related to the task but also rely on knowledge gained through prior experience. The field of Natural Language Processing, [63] makes use of ConceptNet [226] to create richer, contextualized sentence embeddings with the BERT architecture [54]. In [12], the authors utilize the knowledge graph Freebase (now Google Knowledge Graph) to enrich textual representations in a knowledge-based question answering system. In computer vision, [142] exploits commonsense knowledge using Dynamic Memory Networks for Visual Question Answering (VQA), stating it helps the network to reason beyond the image contents. In the scene graph generation task, [84] exploits the ConceptNet [226] knowledge graph to refine object and phrase features to improve the generalization of the model. The authors state that the knowledge surrounding the subject of interest also benefits the inference of objects related to it, helping the model to generalize better and generate meaningful scene graphs. In this work, we exploit commonsense knowledge to enrich a spatial scene representation used for predicting proximity among pairs of objects in a scene context.

3.1.3 (Directed) Spatial Commonsense Graph

The main idea behind our works in object localization [80, 79] is to embed commonsense and geometric knowledge into a scene graph extracted from a partial scan of an area. The scene graphs are then used as problem instances and fed as data to a NN that localizes the object. In this section, we will present the two graph abstractions upon which the NNs are trained: SCG and D-SCG.

Spatial Commonsense Graph (SCG) As illustrated in Fig. 3.2(a), we construct the SCG with nodes that are *i)* object nodes, including all the observed objects in the partially known environment and any target unseen object to be localized, or *ii)* concept nodes that are retrieved from ConceptNet [226]. Each SCG is constructed on top of a Spatial Graph (SG) composed of object nodes that are fully connected. Each object node is further connected to concept nodes via the corresponding semantic relationships. The edges of SCG are of three heterogeneous types:

- *Proximity* relates the pairwise distances between *all* the object nodes given the partial 3D scan;
- *AtLocation* is retrieved from ConceptNet, indicating which environment the objects are often located in;
- *UsedFor* is retrieved from ConceptNet, describing the common use of the objects.

The proximity edges connect all the objects nodes of the SCG in a fully connected manner, while the semantic *AtLocation* and *UsedFor* edges connect each object node with its related concept nodes that are queried from ConceptNet (e.g. *bed AtLocation apartment* or *bed UsedFor resting*). The two semantic edge types provide useful hints on how objects can be clustered in the physical space, thus benefitting the position inference of indoor objects.

Formally, the SCG is an undirected graph composed by a set of nodes $\mathscr{H} = \{h_i | \, i \in (0,N]\}$, where $N = N_o + N_c$ is the total number of nodes in SCG with N_o the number of the object nodes and N_c the number of the concept nodes, and $h_i \in \mathbb{R}^{300}$. Each 300-dimensional vector expresses the node's corresponding word embedding in ConceptNet NumberBatch [227]. The edges are defined by the set $\mathscr{E} = \{e_{ij} | \, i, j \in (0,N], \, i \neq j\}$, where e_{ij} is the edge between node i and node j, and $e_{ij} \in \mathbb{R}^4$. Note that in this formulation, we do not allow for self-loops, as they have little geometric and commonsense meaning for the problem. In practice, we also have an edge signal, represented by a 4-dimensional feature vector whose first three elements indicate the previously explained edge type in a one-hot manner. In contrast, the last element is a scalar, indicating the pairwise distance between two

scene objects. Note that the distance is only measurable on the observed part of the 3D scan (i.e., between known object nodes). Otherwise, we initialise the distance value to -1 when the edges are *AtLocation*, *UsedFor*, or *proximity* edges involving the unknown target object node.

Directed Spatial Commonsense Graph (D-SCG) The directed formulation is very similar to the previous one, i.e., the node set of D-SCG is identical to before. The key difference is that the graph becomes directed by altering the edge signal, which becomes a 6-dimensional feature vector, i.e., $e_{ij} \in \mathbb{R}^6$, whose first three elements indicate the edge type in a one-hot manner, the fourth element indicates whether a proximity relation involves the target node, while the last two elements indicate the relative position $d_{ij} = [\Delta x_{ij}, \Delta y_{ij}]$ between node i and node j, in Cartesian coordinates such that $\Delta x_{ij} = x_j - x_i$ and $\Delta y_{ij} = y_j - y_i$. This definition differs from the SCG, where edges were represented by 4-dimensional vectors representing the one-hot encoded edge class and only the distance between the objects connected by the edge. With this formulation, we can achieve a more data-driven approximation by training in an end-to-end manner. As detailed later in Sec. 3.1.5, this contribution is particularly important for performance improvements in object localization and computational efficiency. Given that the relative positions are only measurable among object nodes in the observed part of the 3D scan, we initialize the relative positions to $[0,0]$ when the edges are *AtLocation*, *UsedFor*, or *Proximity* edges involving the unknown target object node.

3.1.4 SCG Object Localiser (SCG-OL)

First, we describe the two-stage solution used to address the task of localizing the arbitrary unobserved target object using the SCG. In the first stage, we propose a Proximity Prediction Network (PPN) on top of the SCG. PPN aims to predict the pairwise distances between the unseen target object and the objects in the partially known scene. In the second stage, our localization module takes as input the set of pairwise distances, and it outputs the position of the target object based on a probabilistic circular intersection. This solution is, therefore, the one with the most geometric constraints.

Proximity Prediction Network The goal of the PPN is to predict all the pairwise distances between the unseen object and the observed scene objects. We utilize a variant of the Graph Transformer [217] and update the nodes iteratively by doing message passing with attention [243]. Note that this process is a particular instantiation of the general framework presented in Eq.2.4. The procedure is applied between all edge types, allowing for effective fusion between commonsense and geometric knowledge.

The input to the network is the set of node features $\mathscr{H}$ and the output is a new set of node features $\mathscr{H}' = \{h_i' | \, i \in (0,N]\}$, with $h_i' \in \mathbb{R}^D$. Each node i in the graph is updated by aggregating the features of its neighboring nodes $\mathscr{N}_i$ via two rounds of message passing. The resulting h_i' forms a *contextual* representation of its neighborhood.

At each round of message passing, we first learn the attention coefficient α_{ij} using the scaled dot-product attention mechanism [240], conditioned on each edge feature $e_{i,j}$ from node j to node i, and on both nodes' features, h_i and h_j. This allows the network to understand how important each neighbor is for the node representation's update as follows:

$$v_j = W_v h_j + b_v, \tag{3.1}$$

$$\hat{h}_i = \sum_{j \in \mathscr{N}_i} \alpha_{ij}(v_j + e_{i,j}), \tag{3.2}$$

where W_v, b_v represent respectively the weight matrix and bias used to calculate the value vector v for the attention mechanism. The updated node features h_i' are then given by:

$$\tilde{h}_i = (1-\beta_i)\hat{h}_i + \beta_i(W_r h_i + b_r) \tag{3.3}$$

$$h_i' = ReLU(LayerNorm(\tilde{h}_i)), \tag{3.4}$$

where β_i is the output of a gated residual connection [217], which prevents all the nodes from converging into oversmoothed features, W_r, b_r represent the weight matrix and bias respectively used in the linear transformation of h_i. Essentially, the above equations present a learned convex combination of the pre and post attended features, followed by normalization and non-linearity.

After message passing, we obtain the set of final node embeddings $\mathscr{H}^* = \{h_i^* | \, i \in (0,N]\}$, with $h_i^* \in \mathbb{R}^{2D} = (h_i || h'_i)$, where $(\cdot||\cdot)$ represents the concatenation operation. In this way, the final representation of each node contains both the original object embedding and the aggregated embedding of its context in the scene. Finally, we form a novel set of features for each edge between an observed node i and the target node t via concatenation as $h_{it}^* = (h_i^* || h_t^*)$. h_{it}^* is then used by a MLP to predict the pairwise distances $\hat{d}_{it}$ between the target object node t and the observed object node i.

SCG-OL loss To train our PPN, we compute the Mean Square Error (MSE) between the predicted pairwise distances $\hat{d}_{it}$ of the object node i and the target node t and the set of

ground-truth pairwise distances d_{it}. The loss is expressed as:

$$\mathscr{L}_{\text{MSE}}(\hat{d}, d) = \frac{1}{N_o - 1} \sum_{i=1}^{N_o - 1} (\hat{d}_{it} - d_{it})^2. \tag{3.5}$$

Note that the class of the target object can have multiple instances in the unknown part of the scene, i.e., multiple ground-truth positions. Our method, as a localizer, uses the GT position of the instance that is closest to the predicted position for the computation of the MSE loss.

Distances to position: Localisation module In the localization module, we solve the problem of converting the set of predicted object-to-object distances to a single position $\hat{p}_t$ in the space that defines the position of the searched object in a bird's eye view. The distances $\hat{d}_{i,t}$ predicted by the PPN, and the known objects positions p_i, can be used to define a set of circles of radius $\hat{d}_{i,t}$, centered in the positions p_i. With perfect predictions, $\hat{p}_t$ would be obtained as the point of intersection of all the circles. In this case, we would need at least three known object nodes to unambiguously define $\hat{p}_t$. For this reason, in this localization module, we only consider instances with three or more known objects. Let us define $\hat{p}_t$ as the point in the space that minimizes the squared distance from all the circles:

$$\hat{p}_t = \arg\min_{p_t} \sum_{i}^{N_o - 1} (\|p_t - p_i\|_2 - \hat{d}_{i,t})^2. \tag{3.6}$$

While it is possible to obtain a closed form solution of Eq. 3.6 via Linear Least Squares [254], this is not robust to noise in the measured distances, which is likely present in the PPN predictions. An alternative is to minimize this problem by brute force: we first subdivide the space into a grid and compute the sum of the residuals at each position. We then take the position with the lowest value and use it as an initial guess for the Nelder-Mead's simplex algorithm [189] to obtain the final estimate.

3.1.5 D-SCG Object Localiser (D-SCG-OL)

The SCG-OL contains strong geometric priors in that it is invariant w.r.t the Euclidean and Permutation groups. Nevertheless, these priors are enforced by the graph structure, so to validate if we can utilize more correlations in the data, we now present an end-to-end solution based on the directed formulation. In summary, the model first predicts the relative positions of the unseen target object from the objects in the partially known scene. Then, the relative positions are converted into absolute coordinates and mean pooling is applied to estimate the final position. This approach is fully differentiable and requires no additional localization

module based on circular triangulation to predict the position of the target object, thus largely improving our proposed method's efficiency.

Model To predict the relative position of the unseen target node w.r.t. the visible scene objects, we again use a stacked GNN architecture. For this model, our proposed GNN replaces the attention mechanism in Graph Transformer [217] with a sparse attentional message passing mechanism based on ReLA [276]. The idea behind this design choice is that given that the graph is directed, the attention coefficients need not be symmetrical, and therefore we need both sparsity and stronger couplings. The ReLU activation function provides both, given that any negative weights are sparsified and neighbors can be given unbounded positive weights. By doing so, the model can learn very strong correlations present in the dataset. We further add a ScaleNorm and LayerNorm layer to stabilize the training of the new module on our D-SCG. The node embeddings are updated iteratively by utilizing the heterogeneous information of the edge type to allow effective fusion between the commonsense knowledge and the metric measurements. We highlight the main differences between the attention mechanism in [217] and ours in Fig. 3.3.

As with SCG-OL, the input to the network is the set of node features $\mathscr{H}$ and the output is a new set of node features $\mathscr{H}' = \{h'_i | \, i \in (0,N]\}$, with $h'_i \in \mathbb{R}^{300}$. Each node i in the graph is updated by aggregating the features of its neighboring nodes $\mathscr{N}_i$ via four rounds of message passing. The resulting h'_i forms a *contextual* representation of its neighborhood.

At each round of the message passing, we learn an attention coefficient $\alpha_{i,j}$ between each pair of connected nodes using a graph-based and rectified version of the scaled dot-product attention mechanism, conditioned on the node and edge features. Our GNN can learn sparse and (positively) unbounded attention weights due to the usage of the activation function ReLU, as proposed for the vanilla Transformer model by [276], thus allowing for the understanding of arbitrary relationships between the different node types.

The network starts by performing an affine transformation of the relevant node and edge features to calculate the corresponding query, key, value and edge vector that will be used to compute the attention weights:

$$q_i = W_q h_i + b_q, \tag{3.7}$$

$$k_j = W_k h_j + b_k, \tag{3.8}$$

$$v_j = W_v h_j + b_v, \tag{3.9}$$

$$e_{ij} = W_e e_{ij} + b_e, \tag{3.10}$$

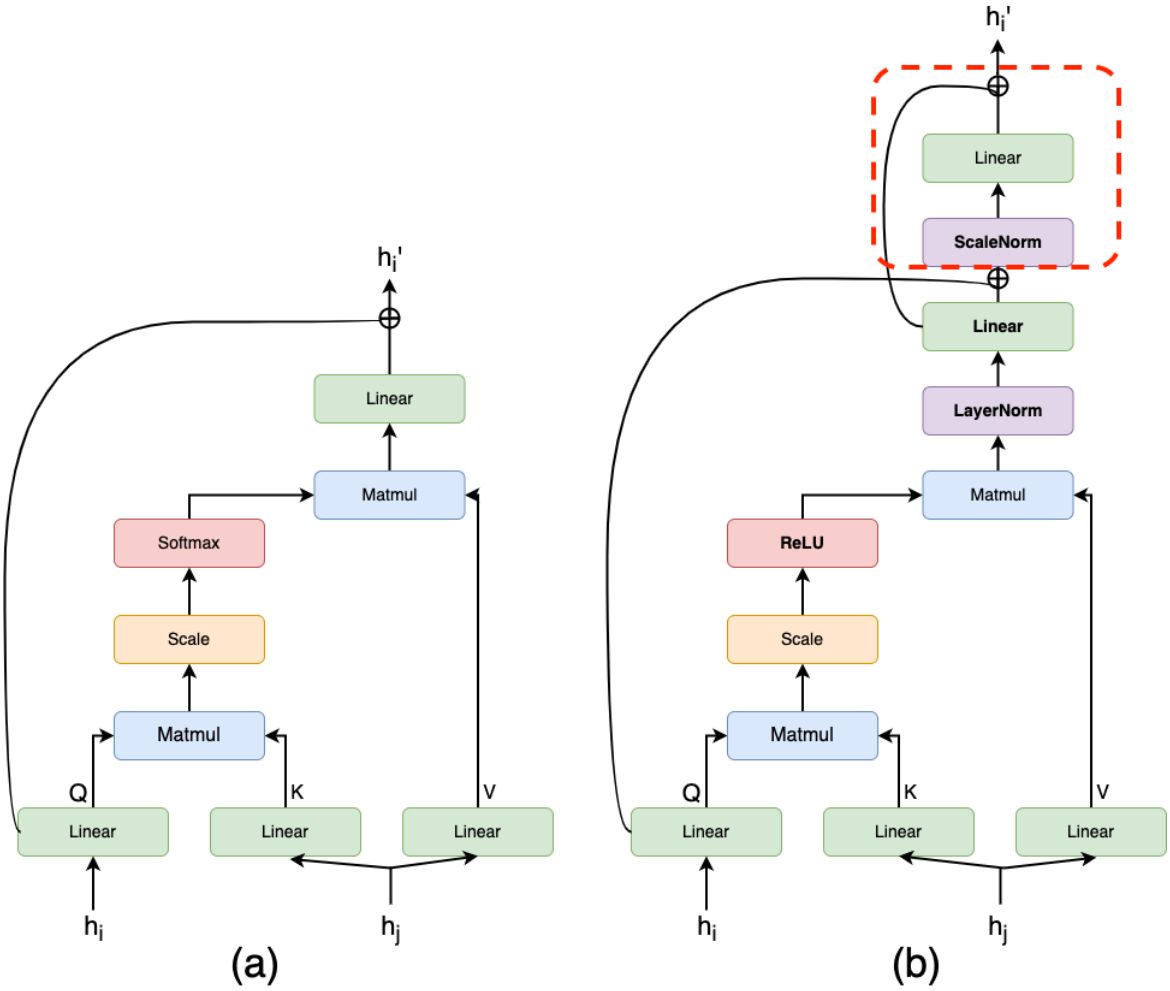


Fig. 3.3 Overview of the differences between the attention mechanism of [217] (a), used in SCG [80], and the one employed in D-SCG Object Localiser, based on ReLA [276] with an added ScaleNorm and reprojection (highlighted in the red dashed box)(b). The new attention mechanism contains more parameters, thus producing more expressive representations, and learns sparse weights with reduced training and inference time thanks to the ReLU activation function. Moreover, we utilize two different normalization layers to stabilize the network's training, given the positively unbounded attention coefficients.

where W and b represent, respectively, the learnable weight matrices and bias vectors for each transformation.

The network then calculates the attention weight $\alpha_{i,j}$ between two nodes i and j as:

$$\alpha_{ij} = ReLU(\frac{\langle q_i, k_j + e_{ij} \rangle}{\sqrt{d}}) \tag{3.11}$$

where $\sqrt{d}$ is a scaling term equal to the square root of the dimension of the projected features k_j. As seen in Eq.3.11, the use of the softmax is dropped since it involves aggregating the scores for all the edges connected to each node, which implies many operations for large graphs and slows down the training. Additionally, using ReLU allows for sparsity in the attention weight matrix, which becomes very useful when analyzing how the network prioritizes the exchange of information. As the attention weights that are calculated using ReLU are not limited to the range $(0,1)$, we use Layer Normalisation [**?**] when calculating the updated node features h'_i, followed by a *gated residual connection* that prevents the node features from converging into indistinguishable features as with SCG-OL:

$$\tilde{h_i} = LayerNorm(h_i + \langle \alpha_{ij}, v_j + e_{ij} \rangle) \tag{3.12}$$

$$\beta_i = Sigmoid(W_g(\tilde{h_i} \,||\, W_{r1}\tilde{h_i} + b_{r1} \,||\, \tilde{h_i} - (W_{r2}\tilde{h_i} - b_{r2}))) \tag{3.13}$$

$$h'_i = (1-\beta_i)\tilde{h_i} + \beta_i(W_{r3}\tilde{h_i} + b_{r3}), \tag{3.14}$$

Differently from [276], we re-normalize and re-project these features similarly to the original Transformer model. This practice has been empirically shown to stabilize and improve the training of self-attentive neural networks [191].

$$h'_i = ScaleNorm(W_o h'_i + b_o), \tag{3.15}$$

This step further increases the number of learnable parameters of our GNN, allowing for better scaling and more expressive representations while not sacrificing efficiency thanks to the sparse attention mechanism (described previously) and the Scale Normalisation in Eq. 3.15. Our sparse graph attention module consists of this sequence of operations, which we use for a total of four message passing rounds. Finally, we obtain the set of final node embeddings $\mathscr{H}^* = \{h^*_i \,|\, i \in (0,N]\}$, with $h^*_i = (h_i \,||\, h'_i)$. In this way, the final representation of each node contains both the original object embedding and the aggregated embedding of its context in the scene. The features of the the target node t and any node i connected to it are concatenated $h^*_{it} = (h^*_i \,||\, h^*_t)$, and the relative position $\hat{d}_{it}$ between the target object node t and the observed object node i is predicted using a linear map. To obtain the final position,

we first convert the relative positions $\hat{d}_{it}$ in absolute coordinates by summing to them the positions $p_i = [x_i, y_i]$ of the observed object nodes and then take the mean of the absolute positions as our predicted position $\hat{p}_t$.

Loss Similarly to the previous setup, we train our network with a strategy that considers that multiple instances of the searched object can exist in the unobserved part of the scene. Therefore, only the instance closest to the prediction is accounted for when calculating the loss. By doing this, the network learns to correctly predict a specific position instead of a point that minimizes the distance w.r.t. all the instances. For the loss, given that we are effectively performing an edge regression task with a scalar target, we aim minimize the squared L2 distance between the predicted position $\hat{p}_t$ and the ground-truth position of the target position p_t as follows:

$$\mathscr{L}_2(\hat{p}_t, p_t) = \|\hat{p}_t - p_t\|_2^2. \tag{3.16}$$

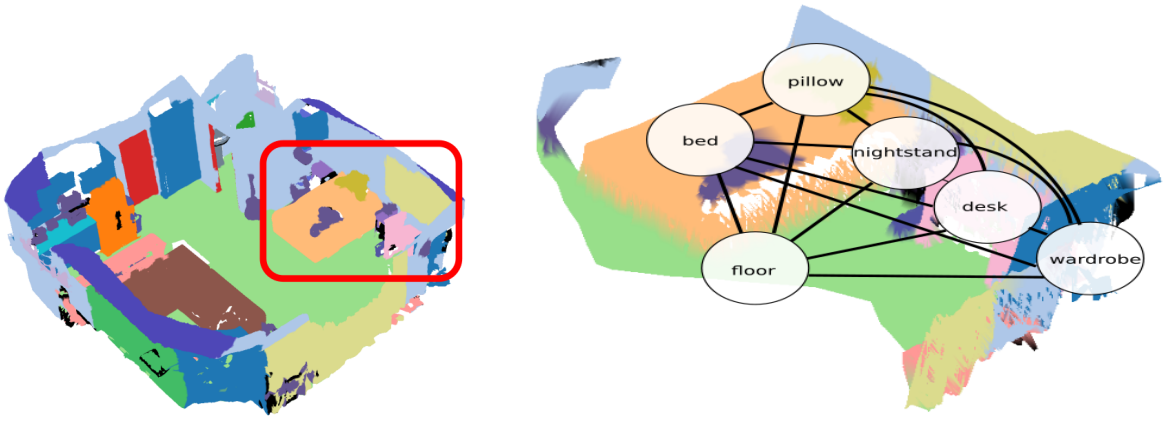
3.1.6 Experiments

Dataset

To validate the proposed models, we build a new dataset of partial 3D scenes using sequences available in ScanNet [48]. ScanNet contains RGB-D sequences taken at a regular frequency with an RGB-D camera. It provides the camera pose corresponding to each captured image and the point-level annotations, i.e., class and instance id, for the complete Point Cloud Data (PCD) of each reconstructed scene. The original acquisition frequency in ScanNet is very high (30Hz), meaning that most images are similar with redundant information for the scene reconstruction. We, therefore, use ScanNet_frames_25k, a subset provided in the ScanNet benchmark[1] with a frequency of about $1/100^{th}$ of the initial one. We further divide the whole RGB-D sequences of each scene into smaller sub-sequences to reconstruct the partial scenes. We vary the length of the sub-sequences to reflect different levels of completeness of the reconstructed scenes. For each sub-sequence, we integrate the RGB-D information with the camera intrinsic and extrinsic parameters to reconstruct the PCD at the resolution of 5cm using Open3D [283]. The annotation for each point in the partial PCD is obtained by looking for the corresponding closest point in the complete PCD scene provided by ScanNet.

From each partially reconstructed scene, we extract the corresponding Spatial Graph with its object nodes, i.e., the graph with only proximity edges (see Fig. 3.4 for an example). The graph nodes contain the object information: e.g., the *position*, defined as the center of the

[1] http://kaldir.vc.in.tum.de/scannet_benchmark



(a) Complete scene (b) Partial scene

Fig. 3.4 The proposed dataset with (a) the complete scene from the ScanNet dataset, and (b) our reconstructed partial scene overlaid with the Spatial Graph.

bounding box containing the object, and the *object class*. We consider the position of each scene object as a 2D point (x, y) on the ground plane, as most objects in the indoor scenes of ScanNet are located at a similar elevation. Each node is marked as *observed* if it represents an object in the partially known scene or as *unseen* if it represents the object in the unknown part of the scene, i.e., the target object to localize.

For the commonsense knowledge, we add the two semantic relationships *AtLocation* and *UsedFor*, as well as the concepts (as nodes) that are linked by the relationships. We extract the concepts from ConceptNet by querying each scene object using the two semantic relationships. The query returns a set of related concepts together with their corresponding weight w, indicating how "safe and credible" each related concept is to the query. We include a concept to the SG only when it has a weight $w > 1$. Fig. 3.5 shows the average number of nodes linked by different types in the (D)SCGs. On average, each (D)SCG contains about 5 times more concept nodes than the object nodes in the (D)SG, demonstrating that rich commonsense knowledge is available. The outliers in the boxplot visualization are introduced by uncommon room types with a large number of objects, e.g., libraries with several books.

Finally, we divide the dataset into training, validation, and testing sets. While we have access to the ScanNet training and validation data (1201 and 312 scenes, respectively), we do not have access to their test data. To address this, we use ScanNet's validation sequences as our testing set and randomly sample a subset of scenes from the training set for validation. By splitting ScanNet's sequences into partial reconstruction, we have 24896 partial scenes with 19461 partial scenes for training and validation and 5435 partial scenes for testing, with each partial scene having its corresponding (D)SCG.

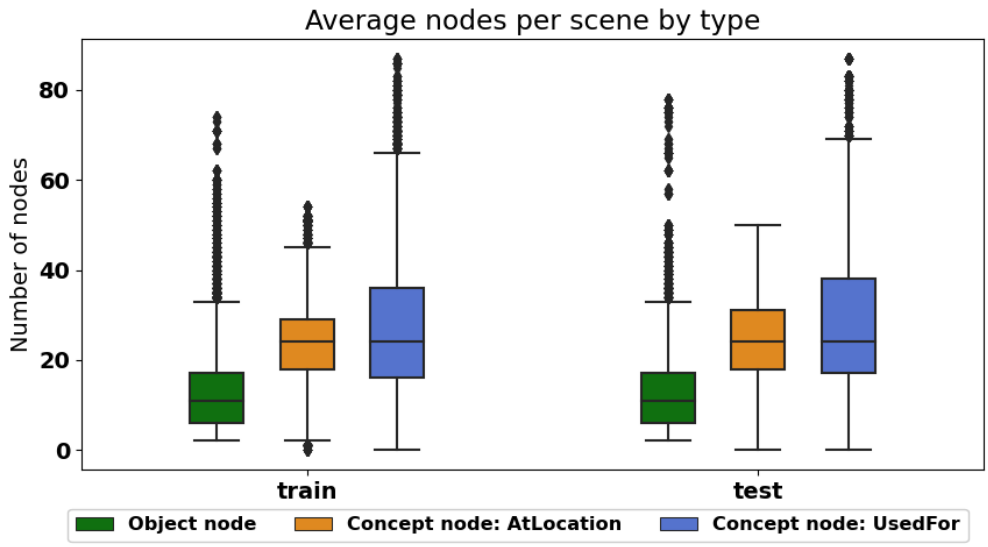


Fig. 3.5 Average number of different types of nodes among the SCGs in the train and test split of the dataset.

Experimental Comparisons

Having introduced the dataset, we can now delve into the in-depth empirical validation that reveals the advantages of our method. First, we provide the implementation details, followed by the metrics used for evaluation. Then, the main body of experiments is comprehensively presented, followed by multiple ablation studies.

Implementation Details. We train our networks using the Adafactor optimiser [214]. The network is trained for 100 epochs. The dimension of the first message passing projection is set to $D = 256$ and $2D$ for the second round. Both use 4 attention heads. For localization with SCG-OL, we ignore edges with a predicted distance of more than 5m, as such high distance values are not trustworthy for localization. For localization with D-SCG-OL, we augment the dataset by applying random rotations to the scene objects during training to compensate for the lack of rotational invariance and allow for better generalization.

Evaluation Measures. We evaluate the performance in terms of both the proximity prediction and target object localization. We quantify the localization performance by the *Localisation Success Rate (LSR)*, defined as the ratio of successful localizations over the number of tests. A localization is considered successful if the predicted position of the target object is close to a target instance within a predefined distance. Unless stated differently, the distance threshold for a positive result is set to 1m. We consider LSR as the *main* evaluation

measure for our task. For the edge proximity prediction, we report the *mean Predicted Proximity Error (mPPE)*, which is the mean absolute error between the predicted distances and the ground-truth pairwise distances between the target object and the objects in the partially known scene. Finally, to quantify the localization accuracy among successful cases, we report the *mean Successful Localisation Error (mSLE)*, which is the mean absolute error between the predicted target position and the ground-truth position among all successful cases.

We compare performance on our new dataset against a set of baselines and state-of-the-art methods for layout prediction. The different methods are evaluated on the localization in 2D (floor plane), given that the height at which an object is located in a room is often not practically relevant. Additional results for 3D localization are reported in the ablation studies. We summarise all the approaches implemented for the comparisons below.

- **Heuristic baselines** use the statistics of the training set, i.e., the *mean*, *mode*, and *median* values of the pairwise distances between the target object and the scene objects, as the predicted distance.

- **MLP** learns to predict pairwise distances between the target object and every other observed object in the scene without considering the spatial or semantic context. The input to this model is a pair of the target object and the observed object, each represented by a one-hot vector indicating the class, passed to an MLP that predicts pairwise distances.

- **MLP with Commonsense** learns to predict the pairwise distance between the target object and every other observed object in the scene without considering the spatial context. We first use GCN to propagate the ConceptNet information to object nodes, and then the features are passed to an MLP that predicts pairwise distances.

- **LayoutTransformer** [88] uses the transformer's self-attention to generate the 2D/3D layout in an auto-regressive manner. We describe the observed objects as a sequence of elements as in [88], where each element contains the object class and the position (x, y). We then feed the target object's class to generate its corresponding position (x, y). For a fair comparison, we retrain the model on our training set.

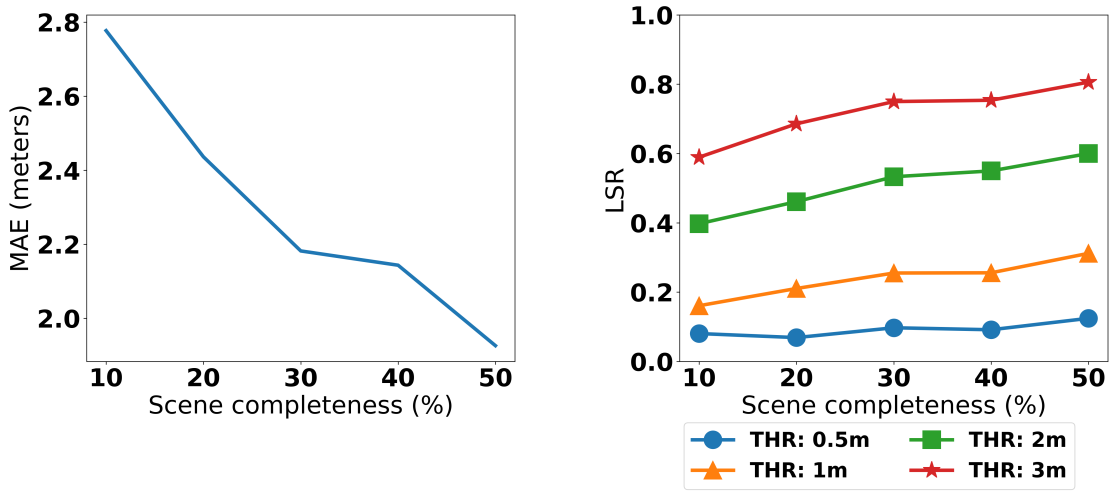
- **SCG-OL** is part of our proposed method, described in Sec. 3.1.4, which exploits SCG. Additional results are presented through a variant trained with learnable node embeddings instead of initializing each node with the commonsense embedding coming from ConceptNet's Numberbatch.

Table 3.1 Methods comparison for object localization in partial scenes. mPPE: mean Predicted Proximity Error. mSLE: mean Successful Localisation Error. LSR: Localisation Success Rate. SG: Spatial Graph. SCG: Spatial Commonsense Graph. D-SG: Directed Spatial Graph. D-SCG: Directed Spatial Commonsense Graph. The first part of the table follows the 2-stage approach, which first predicts the pairwise distances and then localizes the object via multilateration. The last part consists of methods that directly predict the final position.

Method	Data type	mPPE(m)↓	mSLE(m)↓	**LSR ↑**
Statistics-Mean	Pairwise	1.167	0.63	0.140
Statistics-Mode	Pairwise	1.471	0.63	0.149
Statistics-Median	Pairwise	1.205	0.64	0.164
MLP	Pairwise	1.165	0.62	0.143
MLP w/ Commonsense	Pairwise	1.090	0.64	0.163
LayoutTransformer [88]	List	-	**0.59**	0.176
SCG-OL- Learned Emb	SCG	0.974	0.61	0.234
SCG-OL	SCG	0.965	0.61	0.238
Graphormer [267]	D-SCG	-	0.59	0.251
D-SCG-OL w/o Commonsense	D-SG	-	0.59	0.265
D-SCG-OL - Learned Emb	D-SCG	-	0.57	0.273
D-SCG-OL	D-SCG	-	0.55	**0.297**

- **D-SCG-OL w\o Commonsense** is a variant of our approach to test the capability of the method when it is used without commonsense knowledge. The input is the D-SG, composed only of the object nodes and proximity edges. The initial node features are not pre-trained word embeddings but are learned via an embedding layer during training.
- **Graphormer**[267] is an adaptation of Transformer models [240] to graph learning. It uses different embedding strategies to add inductive bias related to the graph structure. It then propagates features between nodes up to k-hop distances using the attention mechanism. We adapted the method so that the node features are updated using the Graphormer network, and then the target node features are used to regress the position of the searched object.
- **D-SCG-OL**is part of our proposed method, described in Sec. 3.1.5, along with a variant that is trained with learnable node embeddings, instead of initializing each node with the commonsense embedding coming from ConceptNet's Numberbatch.

Discussion Table 3.1 reports the localization performance measures in terms of mPPE, LSR, and mSLE, of all compared methods evaluated on the dataset with partially reconstructed scenes. We can initially observe that methods that rely *only* on pairwise inputs,

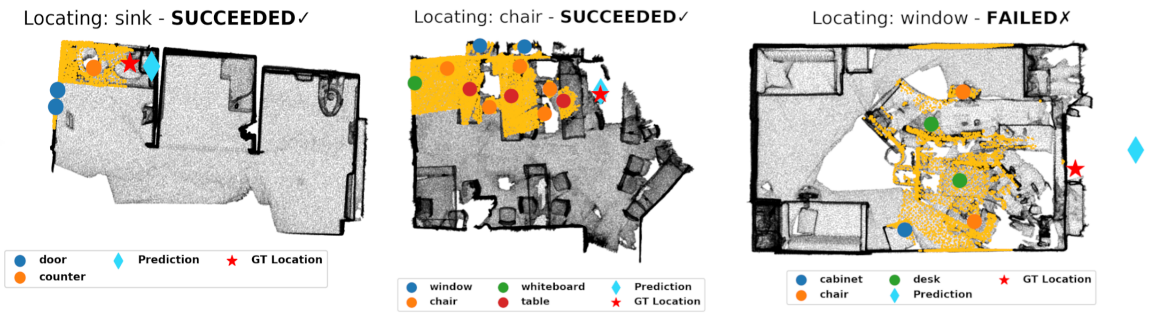


(a) (b)

Fig. 3.6 The proposed dataset with (a) the complete scene from the ScanNet dataset, and (b) our reconstructed partial scene overlaid with the Spatial Graph.

e.g., statistics-based approaches or MLP, lead to worse performance compared to methods that account for other objects present in the observed scene. Nevertheless, introducing semantic reasoning on top of these methods seems to improve the performances, as shown by MLP w/ Commonsense, with an improved LSR of 2% compared to the standard MLP. LayoutTransformer directly predicts the 2D position of the target object by taking as input the list of all the observed scene objects and using the target class as the last input token. It can better encode the spatial context and outperforms the statistic-based and MLP baselines. SCG-OL that uses the SCG with pairwise distances can improve on all metrics w.r.t. the baseline methods, suggesting that a graph-based solution with added geometric and commonsense priors is an effective way of modeling the problem. In this scenario, the addition of pretrained commonsense node embeddings is not extremely important, as shown by the results of SCG-OL- Learned Emb.

When using the DSCG, Graphormer [267] performs best amongst the baselines but fails to reach the same performances as our proposed approach. Graphormer proposes to enhance the propagation of information by aggregating the information not only from directly connected nodes but also from nodes up to a k-distance by creating new edges between them, where the edge features are an inner product between all the edges along the path. While this works fine on a homogeneous graph, the effect can be limited with heterogeneous edge types since the structural information of our proposed heterogeneous graph is not as meaningful as for structures such as molecular graphs, for which Graphormer was proposed. The different versions of our proposed method D-SCG-OL are able to reach



(a) (b) (c)

Fig. 3.7 Qualitative results obtained with D-SCG-OL. The partially known scene is colored with a yellow background, while the unknown scene is indicated with grey. The colored circles indicate the object nodes present in the D-SCG. The *red star* indicates the GT position of the target object, while the *cyan diamond* indicates the predicted positions. The network is able to correctly predict the position of a sink in (a) and a chair in (b). In the failure case of (c), the network correctly identified the direction of the window but overestimated the distance from the visible objects.

the best performance. D-SCG-OL with learned embeddings has a 0.8% increase in the LSR performance w.r.t. the GNN working only on the D-SG, revealing the usefulness of the concept nodes, with a further increase of 2.4% when initializing the node embeddings of the graph by using ConceptNet's Numberbatch, showing that the commonsense information introduced from ConceptNet is helpful for the localization task. Given that it is our best performing method, ablation studies and results from this point forward will only consider D-SCG-OL.

As previously mentioned in this section, we consider the LSR as the primary evaluation metric. It is, therefore, useful to demonstrate and understand how the completeness (visually known) level of the scene impacts the localization performance of D-SCG-OL. Fig. 3.6(a) reports the mean absolute error (MAE) between the estimated position and the ground-truth position compared to the scene completeness. Note that the MAE is calculated on all the test cases, including both the successful and failed ones. For this reason, we use the MAE instead of the already presented mSLE, as the mSLE is calculated only on successful cases and does not vary much with the completeness of the scene. As a general trend, our model can predict more accurately the position of the target object with increasing scene completeness. Fig. 3.6(b) presents how the LSR varies for different localization thresholds as the scene gets more complete. We report the LSR at four different threshold values, i.e., 0.5m, 1m, 2m, and 3m, where a larger threshold leads to a larger LSR value, as expected.

Qualitative results Fig. 3.7 shows the qualitative results obtained using our method D-SCG-OL. Fig. 3.7(a) shows that the "sink" object class was successfully located near the counter. Similarly, in Fig. 3.7(b), the position of the chair (target object) is correctly estimated in a position that is coherent with other instances of chairs and tables in the observed part of the room. Interestingly, Fig. 3.7(c) presents a failure case in which the method fails to locate a window in an office setting. In this case, the network successfully identified the general direction where the window should be located but overestimated its concrete placement with respect to the visible objects. This error is plausible as the network does not effectively see any objects that can help create an idea of the actual shape of the room.

Computational efficiency There are 20.5M parameters in D-SCG-OL, which is 3.4M more than SCG-OL(17.1M). Nevertheless, D-SCG-OL takes 13h35m to fully train the model for 200 epochs on a single Titan RTX, 8x faster than SCG-OL, which requires 108h40m. This large difference is mostly due to the two-stage approach of SCG-OL, which includes the non-differentiable localization module and the more expensive attention mechanism calculations.

Ablation studies We further analyze D-SCG-OL to justify the usefulness of the commonsense relationships and our new attentional message passing mechanism. We also investigate the impact of increasing the number of message passing layers. To verify the applicability in 3D, we also evaluate the localization performance of our method in comparison to the state-of-the-art methods. Lastly, we provide an in-depth investigation of how the attention weights evolve over the message passing when forming the node and edge representation.
Which commonsense relationship is more important? In order to better understand the effects of using different commonsense relationships, we compare the performance of D-SCG-OL against four variants where the D-SCG contains: i) only *Proximity* edges without commonsense relationships, ii) *Proximity* edges with *AtLocation* edges, iii) *Proximity* edges with *UsedFor* edges, and vi) *Proximity* edges with *AtLocation* and *UsedFor* edges. We report the main Localisation Success Rate (LSR) measure for all variants, as well as the scene average percentage of object nodes that are linked by 0, 1, or 2 types of semantic edges, i.e. *AtLocation* and *UsedFor* edges. Table 3.2 shows that *AtLocation* is more effective than *UsedFor* for localizing objects. This is reasonable since the *AtLocation* edge leads to message passing among objects that are connected in the same location, containing information more relevant to the localization task. However, the best performance is obtained when the D-SCG rely on all types of edges which provides a higher connectivity among object nodes to concept nodes. There are 80% object nodes linked to concept nodes by both *AtLocation*

Table 3.2 Impacts of different ConceptNet relationships with the proposed D-SCG-OL. LSR: Localisation Success Rate.

Edge Types	Obj. linked by n semantic edges (%)			LSR ↑
	0	1	2	
Proximity	100	0	0	0.257
AtLocation, Proximity	8	92	0	0.292
UsedFor, Proximity	19	81	0	0.272
AtLocation, UsedFor, Proximity	8	12	80	**0.297**

and *UsedFor* edges, leading to a more effective knowledge fusion than when only one type of semantic edge is used.

Which attention network is more effective? We examine the usefulness of the proposed attention mechanism in D-SCG-OL compared to other, commonly used, attentional message passing modules in the GNN literature. As most of these approaches do not support the use of edge features, we modify the node features for this ablation study to include the positional information to the node features. For a fair comparison, we remove the edge embedding from D-SCG-OL. The set of attention networks we compare with is listed below:

- **No attention [263]** is the first baseline, where we use GraphSAGE without relying on any attention module.
- **GAT [243]** adds an attention mechanism to the message passing procedure.
- **GATv2 [27]** is similar to GAT but improves the attention mechanism in terms of the expressiveness and addresses the problem of "static attention" when using GATs for message passing.
- **HAN [253]** defines multiple meta-paths that connect neighboring nodes either by specific node or edge types. It employs attentional message passing sequentially by first calculating the semantic-specific node embedding and then updating them by using an attention mechanism [240]. With D-SCG we define three sets of meta neighbors, i.e., the proximity neighbours, the *AtLocation* neighbors, and the *UsedFor* neighbors, connected by the specific edges. We implement the message passing for each meta-path using specialized GraphTransformer layers.
- **GraphTransformer [217]** is similar to ours, except that it does not accommodate sparse attention and has less expressive power due to the smaller number of parameters. This module is essentially a porting of the scaled dot-product attention mechanism [240] to GNNs.

Table 3.3 Impacts of different attention modules for the object localisation task with our D-SCG-OL. LSR: Localisation Success Rate.

Graph Attention Type	LSR ↑
No attention [263]	0.199
GAT [243]	0.179
GATv2 [27]	0.202
HAN [253]	0.137
GraphTransformer [217]	0.187
Ours - Only ReLA [276]	0.190
Ours	**0.215**

- **Ours - Only ReLA** [276] is our GNN with the original ReLA without the ScaleNorm and final reprojection layer.

As shown in Table 3.3, different attention modules can produce results that vary greatly in terms of LSR. Among all, HAN achieves the worst performance, showing that features are better to be propagated simultaneously rather than sequentially. GAT and GraphTransformer perform better than HAN, yet it is still worse than GraphSAGE, which uses no attention. This is potentially due to the limitations of the standard attention mechanism when used in GNNs [27]. GraphSAGE is a general inductive framework that leverages node feature information at different depths and is proven to work well on large graphs. In general, the attention module should be carefully designed in order to provide advantageous performance. For example, GATv2 improves the localization performance by fixing the static attention problem of the standard GAT.

Our ReLA-based attention model avoids the usage of a softmax as in the original Graph Transfomer [217] achieves the best overall performance in terms of LSR. This substantial improvement is contributed by the increased expressive power and the ability to reason on sub-graphs during the message passing procedure. The usage of only ReLA, i.e., without scale normalization and the successive projection, achieves a lower LSR compared to the complete module, confirming the advantage brought by the proposed additional normalization as the learned weights are positively unbounded.

How much does the number of message passing layers and the final node concatenation contribute? We examine a set of variants of our D-SCG-OL with varying numbers of message passing layers ranging from 1 to 5. Table 3.4 shows that using four message passing (MP) layers leads to the best performance. When using a single MP layer, there is not enough information regarding the context to be propagated to the nodes and this leads to the worst performance. With more than two MP layers, the performance starts to increase, saturating at four layers. With additional layers, we observe that the performance starts to

Table 3.4 Impact of different numbers of message passing layers in our D-SCG-OL. LSR: Localisation Success Rate.

# Layers	1	2	3	4	5
LSR ↑	0.180	0.257	0.283	**0.297**	0.285

Table 3.5 Comparison of object localization performance in the 3D environment instead of on the 2D floor plane.

Method	LSR ↑
LayoutTransformer [88]	0.158
SCG-OL [80]	0.048
D-SCG-OL	**0.258**

degrade. This might be due to the over-smoothing problem [39, 194], where after multiple message passing rounds, the embeddings for different nodes are indistinguishable from one another. Given the best layer number, we also validate the choice of concatenating the original embedding to the aggregated *contextual* ones instead of using only the aggregated features. Concatenation is more advantageous with an LSR of 0.29, while directly using the aggregated node representation leads to an LSR of 0.28. We argue that this happens because concatenation allows the network to still remember the initial representation, developing a better understanding of the context after message passing.

Localising in 3D. We examine the network capability to localize the target object directly in 3D scenes instead of on the 2D floor plane. We compare D-SCG-OL withSCG-OLby making the appropriate modifications for 3D localization. Table 3.5 reports the localization performance in the 3D scenes. We can observe that all three methods suffer a drop in terms of LSR performance due to the increased difficulty level of the problem. SCG-OL experiences the highest drop in performance, with an LSR score of only 0.05, down from its original score of 0.24 when evaluated in the 2D domain. This decrease in performance can be attributed to a difficulty in representing the object arrangement using only distances when an additional dimension is considered. Utilizing a less abstract representation by using relative positions between objects leads to much more accurate results. Despite the increased problem difficulty, our proposed D-SCG-OL achieves the best LSR of 0.26, which is significantly higher than the second-best method LayoutTransformer with a LSR of 0.15.

Attention Visualisation In Fig. 3.8 and Fig. 3.9, we show how the network prioritizes the exchange of information when localizing a chair. Note that our network does not use the softmax function when calculating the attention weights thus, they do not necessarily sum to one. We normalize the weights for the visualization results. Fig. 3.8 shows the

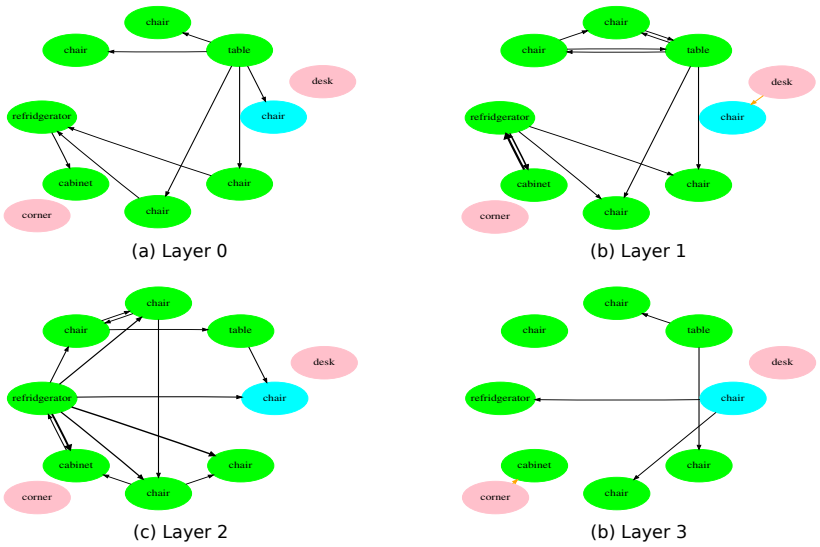


Fig. 3.8 Feature propagation at different layers of our GNN that are directed by our attention module. The cyan node indicates the target object, the green nodes represent the scene nodes, and the pink nodes represent the concept nodes. The black edges indicate the sharing of information between two nodes in the direction indicated by the arrows. For ease of visualization, we show edges with a mean attention weight over the heads that are superior to 0.2%, and only display concept nodes that are connected via these types of edges.

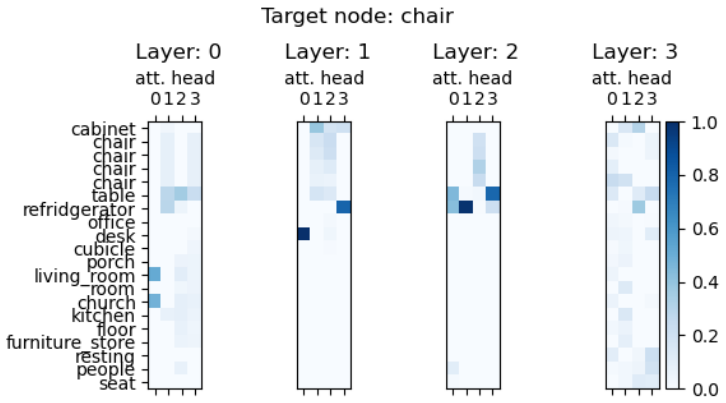


Fig. 3.9 Attention weights for messages that are propagated to the target node are indicated in Fig. 3.8. The network learns to propagate information from different nodes by leveraging different attention heads. The first and last layer of the network propagates information from most of the neighboring nodes, while the intermediate layers focus on a few specific nodes.

features propagated via message passing that are assigned a high weight by our attention modules. The network learns to operate very differently depending on the layer, and most of the attention weights are given to edges between object nodes. The network also learns to attend differently to instances of the same object based on the scene geometry that is described by the edge features. For instance, in the first layer, only two of the five chairs

nodes propagate their features with a high weight to the refrigerator. Incidentally, these nodes represent the two chairs closest to the fridge. Fig. 3.9 shows the different heads' attention scores for messages that are propagated to the target node. We can see that each head focuses on different nodes: some heads are giving high weights to specific nodes, e.g., head zero and three of the second layer, while others balance the features from many nodes, e.g., head two and three of the first layer. Lastly, we can see that most of the commonsense information is propagated in the first and last layer of the GNN.

3.1.7 Conclusions

We addressed the new object localization problem given a partial 3D scan of a scene by proposing a geometrically-inspired graph formulation. We proposed a novel scene graph model, the (Directed) Spatial Commonsense Graph, by augmenting a spatial graph with rich commonsense knowledge to improve the model's inference. With such a graph formulation, we first proposed a two-stage solution for unseen object localization. It initially predicts the pairwise distances between the target node and the other object nodes using the graph-based Proximity Prediction Network. Then, it estimates the target object's position via circular intersection. We also proposed a directed graph formulation, the D-SCG , to address the same challenging problem. The spatial information regarding the arrangement of the object is described via directional edges with relative position vectors instead of the scalar relative distances as in the prior work. With the proposed D-SCG, we developed a new GNN-based solution for object localization, D-SCG Object Localiser, that can directly estimate the position of the target object by predicting its relative positions with respect to other objects in the partially observed scene, leading to an efficient end-to-end trainable solution. Our approach also features a new attention module, w.r.t to our previous approach, to further improve the localization performance by using the ReLA attention. We thoroughly evaluated our proposed method on the partial scene dataset and proved its superior performance in terms of localization success rate against baselines and state-of-the-art methods. Finally, we showed that our approach could be applied for 3D object localization with a marginal performance drop, while the previous state-of-the-art method degrades dramatically due to the increased localization difficulty.

Future work An immediate extension for future work is scaling our proposed approach to larger outdoor scenarios and hierarchical structures such as buildings. Another research avenue is to understand the role of the message passing architecture in the model and if it can be substituted with more advanced flavors, such as those that do graph rewiring [90]. Finally,

from a more applicative perspective, extending this method to robotic applications that can utilize it for object navigation is essential.

Limitations Despite being state-of-the-art, the proposed localization pipeline does not offer a complete solution to this challenging problem. Abstract reasoning over space is a complicated task, and the obtained results reveal that more research is needed before we can adapt object localization in partial scenes in a practical scenario.

Broader impacts Our dataset is built on top of ScanNet, featuring static indoor scenes without the involvement of human subjects. The dataset and the proposed scene graph formulation can facilitate and motivate further research towards visual and geometric scene understanding.

3.2 Human Pose Forecasting in Industrial Scenarios

3.2.1 Introduction

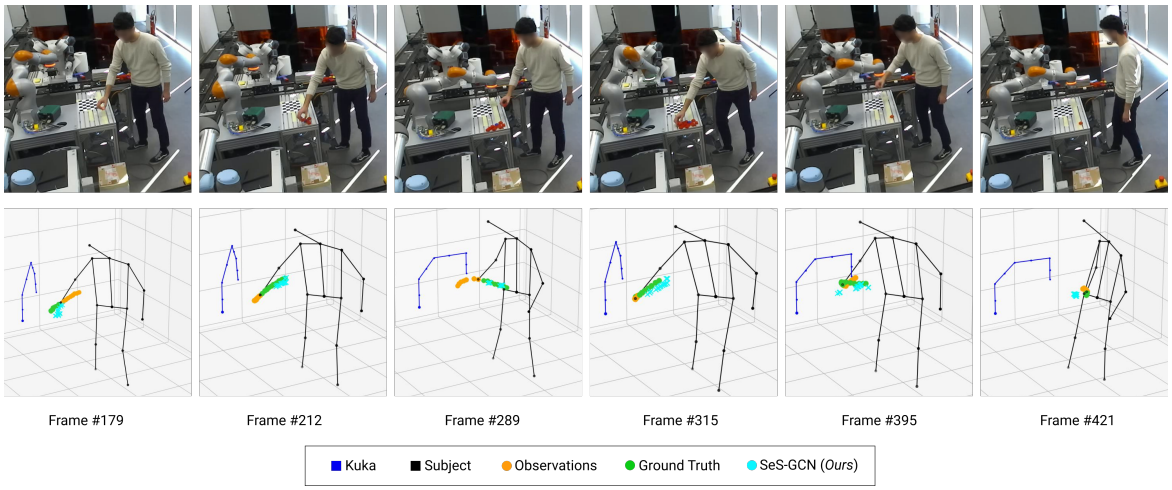


Fig. 3.10 A collision example from our CHICO dataset. On the top row, some frames of the *Lightweight pick and place* action captured by one of the three cameras. On the bottom row are the operator and robot skeletons. The forecasting model takes an observation sequence (in yellow, here pictured for the right wrist only) and performs a prediction (cyan), which is compared with the ground truth (green). In frame 395, it is easy to see the robot hitting the operator, who is retracting, as is evident in frame 421. Note how the predictions by SeS-GCN follow closely the GT, except during the collision. Due to the impact at collision time, the abrupt change of the arm motion produces uncertain predictions, which become extremely difficult to forecast, as shown by the irregular predicted trajectory.

Collaborative robots (cobots) and modern Human-Robot Collaboration (HRC) depart from the traditional separation of functions of industrial robots [127] because of the shared workspace [112]. In this scenario, cobots and humans perform actions concurrently and are likely to engage in physical contact. While there is a clear advantage in increased productivity [206] (improved by as much as 85% [213]) due to the minimization of idle times, there are challenges in the workplace safety [85]: it is not about whether there will be contact, but rather about understanding its consequences [175].

The pioneering work of Shah et al. [213] has already shown that, in order to seamlessly and efficiently interact with human co-workers, cobots need to abide by two collaborative principles: (1) Making decisions on-the-fly, and (2) Considering the consequences of their decision on their teammates. The first calls for promptly and accurately detecting human motion in the workspace. The second principle implies that cobots need to anticipate the pose trajectories of their human co-workers and predict future collisions.

Motivated by these problems, the first contribution of our work is a novel Separable-Sparse Graph Convolutional Neural Network (SeS-GCN) for human pose forecasting. Pose forecasting requires an understanding of the complex spatio-temporal joint dynamics of the human body, and recent trends have highlighted the promises of modeling body kinematics within a single GCN framework [47, 50, 144, 147, 171, 248, 280]. We have designed SeS-GCN with performance and efficiency in mind by bringing together, for the first time, three main modeling principles: depthwise-separable graph convolutions [131], space-time separable graph adjacency matrices [224], and sparse graph adjacency matrices [216]. In SeS-GCN, *separable* stands for limiting the interplay of joints with others (space) at different frames (time) and per channel (depth-wise). Within the GCN, different channels, frames, and joints still interact by means of multi-hop messages. For the first time, sparsity is achieved by a teacher-student framework. The reduced interaction and sparsity results in comparable or fewer parameters than all GCN-based baselines [131, 224, 216], while improving performance by at least 2.7%. Compared to the state-of-the-art (SoA) [170], SeS-GCN is lightweight, using only 1.72% of the parameters, it is $\sim$4 times faster, and remains competitive with just 1.5% larger error on Human 3.6M [111] when predicting 1s in the future. The model is described in detail in Sec. 3.2.3, and experiments with ablation studies are illustrated in Sec. 3.2.5.

We also introduce the very first benchmark of Cobots and Humans in Industrial COllaboration (CHICO, an excerpt in Fig. 3.10). CHICO includes multi-view videos, 3D poses and trajectories of the joints of 20 human operators, in close collaboration with a robotic arm *KUKA LBR iiwa* within a shared workspace. The dataset features 7 realistic industrial actions, taken at a real industrial assembly line with a marker-less setup. The goal of CHICO

is to endow cobots with perceptive awareness to enable human-cobot collaboration with contact. Towards this frontier, CHICO proposes to benchmark two key tasks: human pose forecasting and collision detection. Cobots currently detect collisions by mechanical-only events (transmission of contact wrenches, control torques, sensitive skins). This ensures safety but it harms the human-cobot interaction, because collisions break the motion of both, which reduces productivity, and may be annoying to the human operator. CHICO features 240 1-minute video recordings, from which two separate sets of test sequences are selected: one for estimating the accuracy in pose forecasting, so cobots may be aware of the next future (1.0 sec); and one with 226 genuine collisions, so cobots may foresee them and possibly re-plan. The dataset is detailed in Sec. 3.2.4, and experiments are illustrated in Sec. 3.2.5.

When tested on CHICO, the proposed SeS-GCN outperforms all baselines and reaches an error of 85.3 mm (MPJPE) at 1.00 sec, with a negligible run time of 2.3 msec (as reported in Table 3.10). Additionally, the forecast human motion is used to detect human-cobot collisions by checking whether the predicted trajectory of the human body intersects that of the cobot. This is also encouraging, as SeS-GCN allows to reach an F1-score of 0.64. Both aspects contribute to a cobot awareness of the future, which is instrumental for HRC in industrial applications.

3.2.2 Related Work

Human pose forecasting Human pose forecasting is a recent field that has some intersection with human action anticipation in computer vision [144] and HRC [57]. Previous studies exploited Temporal Convolutional Networks (TCNs) [10, 75, 141, 200] and Recurrent Neural Networks (RNNs) [68, 81, 113]. Both architectures are naturally suited to model the temporal dimension. Recent works have expanded the range of available methods by using Variational Auto-Encoders [32], specific and model-agnostic layers that implicitly model the spatial structure of the human skeleton [4], or Transformer Networks [33].

Pose forecasting using Graph Convolutional Networks (GCN) Most recent research uses GCNs [50, 147, 170, 224, 280]. In [170], the authors have mixed GCN for modeling the joint-joint interaction with Transformer Networks [240] for the temporal patterns. Others [147, 224, 280] have adopted GCNs to model the space-time body kinematics, devising, in the case of [50], hierarchical architectures to model coarse-to-fine body dynamics. We identify three main research directions that have shown efficiency improvements in GCNs:

1. Space-time separable GCNs [224], which factorize the spatial joint-joint and temporal patterns of the adjacency matrix;

Table 3.6 Comparison between the state-of-the-art datasets and the proposed CHICO; *unk* stands for "unknown".

	Quantitative Details							Rec. Scene	Actions Type		Tasks			Markerless
	# Classes	# Subj.	Avg Rec. Time	# Joints	FPS	Aspect Ratio	# Sensors		Industr.	HRC	Action Recog.	Pose Forec.	Coll. Det.	
Human3.6M [111]	15	11	100.49 s	32	25	normalized	15	mo-cap studio				✓		
AMASS [168]	11265	344	12.89 s	variable	variable	original	variable	mo-cap studio				✓		
3DPW [246]	47	7	28.33 s	18	60	original	18	outdoor locations				✓		
ExPI [86]	16	4	*unk*	18	25	original	88	mo-cap studio				✓		
CHI3D [66]	8	6	*unk*	*unk*	*unk*	original	14	mo-cap studio				✓		
InHARD [49]	14	16	< 8 s	17	120	original	20	assembly line	✓	✓	✓			
CHICO(ours)	7	20	55 s	15	25	original	3	assembly line	✓	✓		✓	✓	✓

2. Depth-wise separable graph convolutions [108], which have been explored by [11] in the spectral domain;

3. Sparse GCNs [216], which iteratively prune the terms of the adjacency matrix of a GCN. Notably, all three techniques also yield better performance than the plain GCN.

Here, for the first time, we bring together these three aspects into an end-to-end space-time-depthwise-separable and sparse GCN. The three techniques are complementary to improve both efficiency and performance, but their integration requires some structural changes (*e.g.*, adopting teacher-student architectures for sparsifying), as we describe in Sec. 3.2.3.

Human Robot Collaboration (HRC) HRC is the study of collaborative processes where human and robot agents work together to achieve shared goals [17, 37]. Computer vision studies on HRC are mostly related to pose estimation [35, 73, 140] to locate the articulated human body in the scene. In [40, 118, 187], methodologies for robot motion planning and collision avoidance are proposed; their study perspective is opposite to ours, since we focus on the human operator. In this regard, the works of [19, 46, 119, 150] model the operators' whereabouts through detection algorithms that approximate human shapes using simple bounding boxes. Approaches that predict human motion during collaborative tasks are in [235, 278] using RNNs and in [244] using Gaussian Processes (GPs). Other work [132] models the upper body and the human right hand (which they call the Human End Effector) by considering the robot-human handover phase. As a motion prediction engine, DCT-RNN-GCN [170] is especially considered for the experimental comparisons, given the GCN architecture of the model.

Datasets for pose forecasting Human pose forecasting datasets cover a wide spectrum of scenarios, see Table 3.6 for a comparative analysis. Human3.6M [111] considers everyday actions such as conversing, eating, greeting, and smoking. Data were acquired using a 3D marker-based motion capture system composed of 10 high-speed infrared cameras. AMASS [168] is a collection of 15 datasets where daily actions were captured by an optical marker-based motion capture. Human3.6M and AMASS are standard benchmarks for human pose forecasting, with some overlap in the type of actions they deal with. The 3DPW dataset [246] focuses on outdoor actions captured with a moving camera and 17 Inertial Measurement Units (IMU) embedded on a special suit for motion capturing [202]. The recent ExPI dataset [86] contains 16 different dance actions performed by professional dancers for a total of 115 sequences, and it is aimed at motion prediction. ExPI has been acquired with 68 synchronized and calibrated color cameras and a motion capture system with 20 mocap cameras. Finally, the CHI3D dataset [66] reports 3D data taken from MOCAP systems to study human interactions.

None of these datasets answer our research needs, i.e., a benchmark taken by a sparing, energy-efficient markerless system focused on the industrial HRC scenario, where forecasting may be really useful for anticipating collisions between humans and robots. In fact, the only dataset relating to industrial applications is InHARD [49]. Therein, humans are asked to perform an assembly task while wearing inertial sensors on each limb. The dataset is designed for human action recognition, and it involves 16 individuals performing 13 different actions each, for a total of 4800 action samples over more than 2 million frames. Despite showcasing a collaborative robot, in this dataset the robot is mostly static, making it unsuitable for collision forecasting.

3.2.3 Methodology

We build an accurate, memory-efficient, and fast GCN by bridging three diverse research directions: **i.** Space-time separable adjacency matrices; **ii.** Depth-wise separable graph convolutions; **iii.** Sparse adjacency matrices. This results in an all-separable and Sparse GCN encoder for the human body kinematics, which we dub SeS-GCN, from which the future frames are forecast by a Temporal Convolutional Network (TCN).

Background

Pose forecasting is formulated as observing the 3D coordinates $\boldsymbol{x}_{v,t}$ of V joints across T frames and predicting their location in the K future frames. For convenience of notation, we gather the coordinates from all joints at frame t into the matrix $X_t = [\boldsymbol{x}_{v,t}]_{v=1}^{V} \in \mathbb{R}^{3\times V}$. Then

we define the tensors $\mathbf{X}_{in} = [X_1, X_2 ..., X_T]$ and $\mathbf{X}_{out} = [X_{T+1}, X_{T+2} ..., X_{T+K}]$ that contain all observed input and target frames, respectively. We consider a spatio-temporal graph $\mathbb{G} = (V, E)$ to encode the body kinematics, with all joints at all observed frames as the node set $V = \{\boldsymbol{v}_{i,t}\}_{i=1,t=1}^{V,T}$, and edges $(\boldsymbol{v}_{i,t}, \boldsymbol{v}_{j,s}) \in E$ that connect joints i, j at frames t, s.

Graph Convolutional Networks (GCN) Remember that a GCN is a layered architecture where:

$$\mathscr{X}^{(l+1)} = \sigma\left(A^{(l)} X^{(l)} W^{(l)}\right) \tag{3.17}$$

In this scenario, the input to a GCN layer l is the tensor $X^{(l)} \in \mathbb{R}^{C^{(l)} \times V \times T}$ which maintains the correspondence to the V body joints and the T observed frames, but increases the depth of features to $C^{(l)}$ channels. $X^{(1)} = \mathbf{X}_{in}$ is the input tensor at the first layer, with $C^{(1)} = 3$ channels given by the 3D coordinates. $A^{(l)} \in \mathbb{R}^{VT \times VT}$ is the adjacency matrix relating pairs of VT joints from all frames. Following most recent literature [50, 170, 216, 224], we learn the adjacency matrix $A^{(l)}$ at each GCN layer, which is reported to help model longer interactions. Finally, $W^{(l)} \in \mathbb{R}^{C^{(l)} \times 1 \times 1}$ are the learnable weights of the graph convolutions. σ in our case is defined as the PReLU activation function, which is a generalization of the famous ReLU with a parametrized slope coefficient for negative values [96].

Separable & Sparse Graph Convolutional Networks (SeS-GCN)

We build SeS-GCN by integrating the three aforementioned modeling dimensions: **i.** separating spatial and temporal interaction terms in the adjacency matrix of a GCN; **ii.** separating the graph convolutions depth-wise; **iii.** sparsifying the adjacency matrices of the GCN.

Separating space-time STS-GCN [224] has factored the adjacency matrix $A^{(l)}$ of the GCN, at each layer l, into the product of two terms $A_s^{(l)} \in \mathbb{R}^{V \times V \times T}$ and $A_t^{(l)} \in \mathbb{R}^{T \times T \times V}$, respectively responsible for the temporal-temporal and joint-joint relations. The GCN formulation becomes:

$$\mathscr{X}^{(l+1)} = \sigma\left(A_s^{(l)} A_t^{(l)} X^{(l)} W^{(l)}\right) \tag{3.18}$$

Eq. (3.18) bottlenecks the interplay of joints across different frames, implicitly placing more emphasis on the interaction of joints on the same frame ($A_s^{(l)}$) and on the temporal pattern of each joint ($A_t^{(l)}$). This reduces the memory footprint of a GCN by approximately 4x while improving its performance (cf. Sec. 3.2.5). Note that this differs from alternating spatial and temporal modules, as it is done in [270] and [23], respectively, for trajectory forecasting and action recognition.

Separating depth-wise Inspired by depth-wise convolutions [44, 108], the approach in [131] has introduced depth-wise graph convolutions for image classification, followed by [11], which resorted to a spectral formulation of depth-wise graph convolutions for graph classification. Here, we consider depth-wise graph convolutions for pose forecasting. The depth-wise formulation bottlenecks the interplay of space and time (operated by the adjacency matrix $A^{(l)}$) with the channels of the graph convolution $W^{(l)}$. The resulting all-separable model, which we dub STS-DW-GCN, is formulated as such:

$$\mathscr{H}^{(l)} = \gamma\left(A_s^{(l)} A_t^{(l)} X^{(l)} W_{DW}^{(l)}\right) \tag{3.19a}$$

$$X^{(l+1)} = \sigma\left(\mathscr{H}^{(l)} W_{\text{MLP}}^{(l)}\right) \tag{3.19b}$$

Adding the depth-wise graph convolution splits the GCN of layer l into two terms. The first, Eq. (3.19a), focuses on space-time interaction and limits the channel cross-talk by the use of $W_{DW}^{(l)} \in \mathbb{R}^{\frac{C^{(l)}}{\alpha} \times 1 \times 1}$, with $1 \leq \alpha \leq C^{(l)}$ setting the number of convolutional groups ($\alpha = C^{(l)}$ is the plain single-group depth-wise convolution). Eq. (3.19b) simply models the intra-channel communication. This may be understood as a plain (MLP) 1D-convolution with $W_{\text{MLP}}^{(l)} \in \mathbb{R}^{C^{(l)} \times 1 \times 1}$ which re-maps features from $C^{(l)}$ to $C^{(l+1)}$. γ is the ReLU6 non-linear activation function. Overall, this does not significantly reduce the number of parameters, but it deepens the GCN without over-smoothing [195], which improves performance (see Sec. 3.2.5 for details).

Sparsifying the GCN Sparsification has been used to improve the efficiency (memory and, in some cases, runtime) of neural networks since the seminal pruning work of [133]. [216] has sparsified GCNs for trajectory forecasting. This consists in learning masks $\mathscr{M}$ which selectively erase certain parameters in the adjacency matrix of the GCN. Here we integrate sparsification with the all-separable GCN design, which yields our proposed SeS-GCN for human pose forecasting:

$$\mathscr{H}^{(l)} = \gamma\left((M_s^{(l)} \odot A_s^{(l)})(M_t^{(l)} \odot A_t^{(l)}) X^{(l)} W_{DW}^{(l)}\right) \tag{3.20a}$$

$$X^{(l+1)} = \sigma\left(\mathscr{H}^{(l)} W_{\text{MLP}}^{(l)}\right) \tag{3.20b}$$

$\odot$ is the element-wise product and $M_{\{s,t\}}^{(l)}$ are binary masks. Both at training and inference, [216] generates masks, it uses those to zero certain coefficients of the adjacency matrix A, and it adopts the resulting GCN for trajectory forecasting. By contrast, we adopt a teacher-student framework during training. The teacher learns the masks, and the student only considers the spared coefficients in A. At inference, our proposed SeS-GCN only consists of the

student, which simply adopts the learnt sparse A_s and A_t. Compared to [216], the approach of SeS-GCN is more robust at training, it yields fewer model parameters at inference ($\sim$30% less for both A_s and A_t), and it reaches a better performance, as it is detailed in Sec. 3.2.5.

Decoder Forecasting

Given the encoded space-time representation by SeS-GCN, the future frames are then decoded by using a Temporal Convolutional Network (TCN) [10, 75, 141, 224]. The TCN remaps the temporal dimension to match the sought output number of predicted frames. We do not consider this part for further development as it offers a great tradeoff in parameter efficiency vs performance, and it is a consolidated practice in current approaches to pose forecasting.

3.2.4 The CHICO dataset

In this section, the CHICO dataset is detailed by describing the acquisition scenario and devices, the cobot and the performed actions. We have rendered RGB videos, skeletons and calibration parameters public in order to help facilitate future research on this topic. [2].

The scenario The dataset is registered in a smart-factory environment, with a single human operator standing in front of a $0.9\,\text{m} \times 0.6\,\text{m}$ workbench and a cobot at its end (see Fig.3.10). The human operator has some free space to turn towards some equipment and carry out certain assembly, loading, and unloading actions [180]. In particular, light plastic pieces and heavy tiles, a hammer, and abrasive sponges are available. A total of 20 human operators have been hired for this study. They attended a course on how to operate with the cobot and signed an informed consent form prior to the recordings.

The collaborative robot A 7 degrees-of-freedom Kuka LBR iiwa 14 R820 collaborates with the human operator during the data acquisition process. Weighing in at 29.5 kg and with the ability to handle a payload up to 14 kg, it is widely used in modern production lines. More details on the cobot can be found in the supplementary material.

The acquisition setup The acquisition system is based on three RGB HD cameras providing three different viewpoints of the same workplace: two frontal-lateral and one rear view. The frame rate is 25Hz. The videos were first checked for erroneous or spurious frames, then we used Voxelpose [236] to extract the 3D human pose for each frame. Extrinsic parameters of each camera are estimated w.r.t. the robot's reference frame by means of a

[2]Code and dataset are available at: https://github.com/AlessioSam/CHICO-PoseForecasting.

calibration chessboard of 1×1m, and temporal alignment is guaranteed by synchronization of all the components with an Internet Time Server. In our environment, Voxelpose estimates a joint positioning accuracy in terms of Mean Per Joint Position Error (MPJPE) of 24.99mm using three cameras, which is enough for our purposes as an ideal compromise between the portability of the system and accuracy. We confirm these numbers in two ways: the first is by checking that human-cobot collisions were detected with 100% F1 score (we have a collision when the minimum distance between the human limbs and the robotic links is below a predefined threshold). Secondly, we show that the new CHICO dataset does not suffer from a trivial zero velocity solution [172], i.e., results achieved by a zero velocity model underperform the current SoA in equal proportion as for the large-scale established Human3.6M.

Actions The 7 types of actions of CHICO are inspired by ordinary work sessions in an HRC disassembly line as described in the review work of [105]. Each action is repeated over a time interval of $\sim$1 minute on average. Each action is associated with a goal that the human operator has to achieve by a given time limit, which requires them to move with a certain velocity. Each action consists of repeated interactions with the robot (*e.g.*, robot place, human picks) which, due to the limited space, lead to some *unconstrained collisions*[1] which we label accordingly. Globally, from the 7 actions $\times$ 20 operators, we collect 226 different collisions. In the following, each action is briefly described.

- ***Lightweight pick and place*** (*Light P&P*). The human operator is required to move small objects of approximately 50 grams from a loading bay to a delivery location within a given time slot. The bay and the delivery location are at the opposite sides of the workbench. Meanwhile, the robot loads on of this bay so that the human operator has to pass close to the robotic arm. In many cases, the distance between the limbs and the robotic arm is a few centimeters.

- ***Heavyweight pick and place*** (*Heavy P&P*). The setup of this action is the same as before, but the objects to be moved are floor tiles weighing 0.75 kg. This means that the actions have to be carried out with two hands.

- ***Surface polishing*** (*Polishing*). This action was inspired by [167], where the human operator polishes the border of a 40 by 60cm tile with some abrasive sponge, and the robot mimics a visual quality inspection.

[1] Unconstrained collisions is a term coming from [91], indicating a situation in which only the robot and human are directly involved into the collision.

- ***Precision pick and place*** (*Prec. P&P*). The robot places four plastic pieces in the four corners of a 30×30cm table in the center of the workbench, and the human has to remove them and put them on a bay before the robot repeats the same unloading.
- ***Random pick and place*** (*Rnd. P&P*). Same as the previous action, except for the plastic pieces, which were continuously placed by the robot randomly on the central 30×30cm table, and the human operator had to remove them.
- ***High shelf lifting*** (*High lift*). The goal was to pick light plastic pieces (50 grams each) on a sideway bay filled by the robot, putting them on a shelf located at 1.70m, on the opposite side of the workbench. Due to the geometry of the workspace, the arms of the human operator were required to pass above or below the moving robotic arm. In this way, close distances between the human arm and forearm and the robotic links were realized.
- ***Hammering*** (*Hammer*). The operator hits a metallic tide with a hammer held by the robot. In this case, the interest was to check how much the collision detection is robust to an action where the human arm is colliding close to the robotic arm (that is, on the metallic tile) without properly colliding *with the robotic arm*.

3.2.5 Experiments

Experiments on Human3.6M

We benchmark the proposed SeS-GCN model on the large and established Human3.6M [111]. In Sec. 3.2.5, we analyze the design choices corresponding to the models discussed in Sec. 3.2.3, then we compare with the state-of-the-art in Sec. 3.2.5.

Human3.6M [111] is an established dataset for pose forecasting, consisting of 15 daily life actions (e.g. Walking, Eating, Sitting Down). From the original skeleton of 32 joints, 22 are sampled as the task, representing the body kinematics. A total of 3.6 million poses are captured at 25 fps. In line with the literature [170, 172, 50], subjects 1, 6, 7, 8, 9 are used for for training, subject 11 for validation, and subject 5 for testing.

Metric The prediction error is quantified via the MPJPE error metric [111, 171], which considers the displacement of the predicted 3D coordinates w.r.t. the ground truth, in millimeters,

at a specific future frame t:

$$L_{\text{MPJPE}}(\hat{\boldsymbol{x}}, \boldsymbol{x}) = \frac{1}{V} \sum_{v=1}^{V} ||\hat{\boldsymbol{x}}_{vt} - \boldsymbol{x}_{vt}||_2. \tag{3.21}$$

where $\hat{\boldsymbol{x}}$ represents the predictions of the model for all v joints and $\boldsymbol{x}$ the corresponding ground truth.

Modelling choices of SeS-GCN

We review and quantify the impact of the modeling choices of SeS-GCN:

Efficient GCN baselines In Table 3.7, we first validate the three difference modeling approaches to efficient GCNs, namely space-time separable STS-GCN [224], depth-wise separable graph convolutions DW-GCN [131], and Sparse-GCN [216]. STS-GCN yields the lowest MPJPE error of 117.0 mm at a 1s forecasting horizon (2.4% better than DW-GCN, 4.8% better than Sparse-GCN) with the fewest parameters, 57.6k (ca. x4 less). We build, therefore, on this approach.

Deeper GCNs. It is a long standing belief that Deep Neural Networks (DNN) owe their performance to depth [97, 154, 259, 274]. However, deeper models require more parameters and have a longer processing time. Additionally, deeper GCNs may suffer from over-smoothing [195]. Seeking both better accuracy and efficiency, we consider three pathways for improvement: (1) add GCN layers; (2) add MLP layers between layers of GCNs; (3) adopt depth-wise graph convolutions, which also add MLP layers between GCN ones (cf. Sec. 3.2.3). As shown in Table 3.7, there is a slight improvement in performance with 5

Table 3.7 MPJPE error (millimeters) for long-range predictions (25 frames) on Human3.6M [111] and numbers of parameters. Best figures overall are reported in bold, while underlined figures represent the best in each block. The proposed model has comparable or less parameters than the GCN-based baselines [108, 216, 224] and it outperforms the best of them [224] by 2.6%.

	Depth	MPJPE	Parameters (K)	DW-Separable	ST-Separable	Sparse	w/ MLP layers	Teacher-Student
GCN	4	123.2	222.7					
DW-GCN [131]	4+4	119.8	223.2	✓			✓	
STS-GCN^2 [224]	4	117.0	**57.6**		✓			
Sparse-GCN [216]	4	122.7	257.9			✓		
STS-GCN	5	115.9	68.6		✓			
STS-GCN	6	116.1	79.9		✓			
STS-GCN w/ MLP	5+5	125.2	101.4		✓		✓	
STS-DW-GCN	5+5	114.8	70.0	✓	✓		✓	
STS-DW-Sparse-GCN	5+5	115.7	122.4	✓	✓	✓	✓	
SeS-GCN (proposed)	5+5	**113.9**	58.6	✓	✓	✓	✓	✓

STS-GCN layers (MPJPE of 115.9 mm), but deeper models underperform. Adding MLP layers between the GCN ones (depth of 5+5) also decreases performance (MPJPE of 125.2). By contrast, adding depth by depth-wise separable graph convolutions (STS-DW-GCN of depth 5+5) reduces the error to 114.8 mm. This may be explained by the virtues of the increased depth in combination with the limiting cross-talk of joint-time channels, which existing literature confirms [44, 131, 224]. We note that space-time and depth-wise channel separability are complementary. Altogether, this performance is beyond the STS-GCN performance (114.8 Vs. 117.0 mm), at a slight increase of the parameter count (70k Vs. 57.6k).

Sparsifying GCNs and the proposed SeS-GCN Finally, we target to improve efficiency by model compression. Trends have reduced the size of models by reducing the parameter precision [203], by pruning and sparsifying some of the parameters [182], or by constructing teacher-student frameworks, whereby a smaller student model is paired with a larger teacher to reach its same performance [101, 145]. Note that the last technique is the current go-to choice in deploying very large networks such as Transformers [20]. We start off by compressing the model with sparse adjacency matrices by the approach of Sparse-GCN [216]. They iteratively optimize the learned parameters and the masks to select some (the selection occurs by a network branch, also at inference, cf. 3.2.3). As illustrated in Table 3.7, the approach of [216] does not make a viable direction (STS-DW-Sparse-GCN), since the error increases to 115.7 mm and the parameter count to 122.4k. Reminiscent of teacher-student models, in the proposed SeS-GCN we first train a teacher STS-DW-GCN, then use its learned parameters to sparsify the affinity matrices of a student STS-DW-GCN, which is then trained from scratch. SeS-GCN achieves a competitive parameter count and the lowest MPJPE error of 113.9 mm, being comparable with the current SoA [170] and using only 1.72% of its parameters (58.6k Vs. 3.4M).

Comparison with the state-of-the-art (SoA) In Table 3.8, we evaluate the proposed SeS-GCN against the three most recent techniques over a short time horizon (10 frames, 400 msec) and a long time horizon (25 frames, 1000 msec). The first, DCT-RNN-GCN [170], the current SoA, uses DCT encoding, motion attention and RNNs and, differently from other models, demands more frames as input (50 vs. 10). The other two, MSR-GCN [50] and STS-GCN [224] adopt GCN-only frameworks, the former adopts a multi-scale approach, the latter acts a separation between spatial and temporal encoding. Both on short- and long-term predictions, at the 400 and 1000 msec horizons, the proposed SeS-GCN outperforms other

techniques [224, 170] and it is within a 1.5% error w.r.t. the current SoA [170], while only using 1.72% parameters and being ∼4 times faster than [170].

Table 3.8 MPJPE error in mm for short-term (400 msec, 10 frames) and long-term (1000 msec, 25 frames) predictions of 3D joint positions on Human3.6M. The proposed model achieves competitive performance with the SoA [170], while adopting 1.72% of its parameters and running ∼4 times faster, cf. Table 3.10. Results are discussed in Sec. 3.2.5.

	Walking		Eating		Smoking		Discussion		Directions		Greeting		Phoning		Posing	
Time Horizon (msec)	*400*	*1000*	*400*	*1000*	*400*	*1000*	*400*	*1000*	*400*	*1000*	*400*	*1000*	*400*	*1000*	*400*	*1000*
DCT-RNN-GCN [170]	**39.8**	**58.1**	**36.2**	**75.5**	**36.4**	**69.5**	**65.4**	119.8	56.5	106.5	**78.1**	138.8	**49.2**	105.0	**75.8**	178.2
MSR-GCN [50]	45.2	63.0	40.4	77.1	38.1	71.6	69.7	117.5	**53.8**	**100.5**	93.3	147.2	51.2	**104.3**	85.0	174.3
STS-GCN^2 [224]	51.0	70.2	43.3	82.6	42.3	76.1	71.9	118.9	63.2	109.6	86.4	**136.1**	53.8	108.3	84.7	178.4
SeS-GCN (proposed)	48.8	67.3	41.7	78.1	40.8	73.7	70.6	**116.7**	60.3	106.9	83.8	137.2	52.6	106.7	82.6	**173.5**

	Purchases		Sitting		Sitting Down		Taking Photo		Waiting		Walking Dog		Walking Together		Average	
Time Horizon (msec)	*400*	*1000*	*400*	*1000*	*400*	*1000*	*400*	*1000*	*400*	*1000*	*400*	*1000*	*400*	*1000*	*400*	*1000*
DCT-RNN-GCN [170]	**73.9**	**134.2**	**56.0**	**115.9**	**72.0**	**143.6**	**51.5**	**115.9**	**54.9**	108.2	**86.3**	**146.9**	**41.9**	**64.9**	**58.3**	**112.1**
MSR-GCN [50]	79.6	139.1	57.8	120.0	76.8	155.4	56.3	121.8	59.2	**106.2**	93.3	148.2	43.8	65.9	62.9	114.1
STS-GCN^2 [224]	83.1	141.0	60.8	121.4	79.4	148.4	59.4	126.3	62.0	113.6	97.3	151.5	49.1	72.5	65.8	117.0
SeS-GCN (proposed)	82.2	139.1	59.9	117.5	78.1	146.0	57.7	121.2	58.5	107.5	94.0	147.7	48.3	70.8	64.0	113.9

Experiments on CHICO

We benchmark on CHICO the SoA and the proposed SeS-GCN model. The first part reports results on the task of human pose forecasting, while Sec. 3.2.5 discusses collision detection.

Pose forecasting benchmark and evaluation protocol We create the train/validation/test split by assigning 2 subjects to the validation (subjects 0 and 4), 4 to the test set (subjects 2, 3, 18 and 19), and the remaining 14 to the training set. For short-range prediction experiments, abiding the setup of Human3.6M [111], we consider 10 frames as observation time and 10 or 25 frames as forecasting horizon. Differently from all reported techniques, DCT-RNN-GCN requires 50 input frames. We adopt the same Mean Per Joint Position Error (MPJPE)[111] as Human3.6M, in Eq. (3.21), which also defines the training loss for the evaluated techniques. None of the motion sequences for pose forecasting contains collisions. In fact, the objective is to train and test the "correct" collaborative human behavior, and not the human retractions and the pauses due to the collisions[7].

Comparative evaluation. In Table 3.9, we compare pose forecasting techniques from the SoA and the proposed SeS-GCN. On the short-term predictions the best performance is that

[7]After the collisions, the robot stops for 1 seconds, during which the human operator usually stands still, waiting for the robot to resume operations.

Table 3.9 MPJPE error in mm for short-term (400 msec, 10 frames) and long-term (1000 msec, 25 frames) prediction of 3D joint positions on CHICOdataset. The average error is 7.9% lower than the other models in the short-term and 2.4% lower in the long-term prediction. See Sec. 3.2.5 for a discussion.

	Hammer		High Lift		Prec. P&P		Rnd. P&P		Polishing		Heavy P&P		Light P&P		Average	
Time Horizon (msec)	*400*	*1000*	*400*	*1000*	*400*	*1000*	*400*	*1000*	*400*	*1000*	*400*	*1000*	*400*	*1000*	*400*	*1000*
DCT-RNN-GCN [170]	41.1	**39.0**	69.4	128.8	50.6	83.3	52.7	88.2	42.1	76.0	64.1	121.5	62.1	104.2	54.6	91.6
MSR-GCN [50]	41.6	39.7	67.8	130.2	50.2	81.3	53.4	90.3	41.1	73.2	62.7	118.2	61.5	101.9	54.1	90.7
STS-GCN [224]	46.6	52.1	64.2	116.4	48.3	79.5	52.0	87.9	42.1	73.9	60.6	106.5	57.2	95.2	53.0	87.4
SeS-GCN (proposed)	**40.9**	49.3	**62.1**	**116.3**	**46.0**	**77.4**	**48.4**	**84.8**	**38.8**	**72.4**	**56.1**	**104.4**	**56.2**	**92.2**	**48.8**	**85.3**

of SeS-GCN, reaching an MPJPE error of 48.8 mm, which is 7.9% better than the second best STS-GCN [224]. On the longer-term predictions, the best performance (MPJPE error of 85.3 mm) is also detained by SeS-GCN, which is 2.4% better than the second best STS-GCN [224]. The proposed model outperforms all techniques on all actions except *Hammer*, a briefly repeating action which may differ for single hits. We argue that DCT-RNN-GCN [170] may get an advantage from using 50 input frames (all other methods use 10 frames)

For a graphical illustration, Fig. 3.11 shows a distribution of the error per joint calculated over all the actions, for the horizons 400 (*left*) and 1000 msec (*right*). In both cases the error gets larger as we get closer to the extrema of the kinematic skeleton, since those joints move the most. The slightly larger error at the right hands (70.03 and 125.76 mm, respectively) matches that subjects are right-handed (but some actions are operated with both hands). For a sanity check of results, we have also evaluated the performance of a trivial zero velocity model. [172] has found that keeping the last observed positions may be a surprisingly strong (trivial) baseline. For CHICO, the zero velocity model scores an MPJPE of 110.6 at

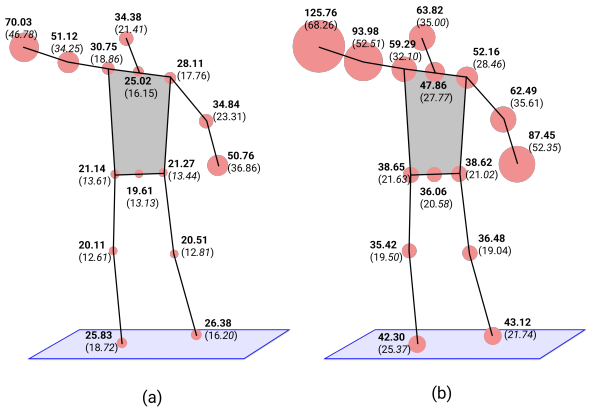


Fig. 3.11 Average MPJPE distribution for all actions in CHICOon different joints for (a) short-term (0.40 s) and (b) long-term (1.00 s) predictions. The radius of the blob gives the spatial error with the same scale of the skeleton.

25-frames, worse than the 85.3 mm score of SeS-GCN. This is in line with the large-scale dataset Human3.6M [111], where the performance of the trivial model is 153.3 mm.

Collision detection evaluation protocol We consider a collision to occur when any body limb of the subject gets too close to any part of the cobot, i.e. within a distance threshold, for at least one frame. In particular, a collision refers to the proximity between the cobot and the human in the forecast portion of the trajectory. The (Euclidean) distance threshold is set to 13 cm. The motion of the cobot is scripted beforehand, thus known. The motion of the human subjects in the next 1000 msec needs to be forecast, starting from the observation of 400 msec. The train/validation/test sets sample sequences of 10+25 frames with stride of 10.

Evaluation of collision detection. For the evaluation of collision, following [181], both the cobot arm parts and the human body limbs are approximated by cylinders. The diameters for the cobot are fixed to 8cm. Those of the body limbs are taken from a human atlas. In Table 3.10, we report precision, recall and F_1 scores for the detection of collisions on the motion of 2 test subjects, which contains 21 collisions. The top performer in pose forecasting, our proposed SeS-GCN, also yields the largest F_1 score of 0.64. The lower performing MSR-GCN [50] yields poor collision detection capabilities, with an F_1 score of 0.31.

Table 3.10 Evaluation of collision detection performance achieved by competing pose forecasting techniques, with indication of inference run time. See discussion in Sec. 3.2.5.

Time Horizon (msec)	***1000***			
Metrics	*Prec*	*Recall*	F_1	*Inference Time (sec)*
DCT-RNN-GCN [170]	0.63	0.58	0.56	9.1×10^{-3}
MSR-GCN [50]	0.63	0.30	0.31	25.2×10^{-3}
STS-GCN [224]	0.68	0.61	0.63	$\mathbf{2.3 \times 10^{-3}}$
SeS-GCN (proposed)	0.84	0.54	**0.64**	$\mathbf{2.3 \times 10^{-3}}$

3.2.6 Conclusions

Towards the goal of forecasting the human motion during human-robot collaboration in industrial (HRC) environments, we have proposed the novel SeS-GCN model, which integrates three most recent modelling methodologies for accuracy and efficiency: space-time separable GCNs, depth-wise separable graph convolutions and sparse GCNs. Also, we have contributed a new CHICO dataset, acquired at real assembly line, the first providing a benchmark of the two fundamental HRC tasks of human pose forecasting and collision detection. Featuring an MPJPE error of 85.3 mm at 1 sec in the future with a negligible run time of 2.3 msec,

SeS-GCN and CHICO unleash great potential for perception algorithms and their application in robotics.

3.3 Chapter takeaways

In this chapter, we explored how learning on graphs through geometric priors and learning on 3D geometric data helps solve prominent CV problems. Both object localization (Sec. 3.1) and human pose forecasting (Sec. 3.2) have real and practical impact, especially when considering robotics applications in industrial scenarios. On one hand, object localization in partial scenes can provide useful prior knowledge for the purpose of navigation, making it goal driven. On the other hand, as mentioned numerous times in the previous section, human pose forecasting is very important in Industry 4.0 scenarios. Therefore, both of the proposed works can facilitate the use of robotic assistants in manufacturing plants.

Nevertheless, both works presented in this section also have important scientific takeaways. Firstly, following the large body of work in Geometric Deep Learning, we see that the data efficiency of methods that rely on geometric priors is much higher than simply trying to "blindly" learn everything in a data-driven way. This becomes evident in the performance vs dataset size tradeoff between SCG and D-SCG in Sec. 3.1. Secondly, the methods we propose based on graph neural networks seem to offer ways of speeding up computation that are very intuitive to implement (sparsification, adjacency separation). Graph structures also offer a way to abstract problems without having to always process the actual domain, which in this case is visual. Unlike other alternatives such as 3D Convolutions, working with a graph formulation is much more efficient and can generalize across datasets, given that it ignores details related to scene volume or colors.

With the completion of this chapter, the subsequent sections will explore the second forking path of the thesis. We will delve into fundamental aspects of learning techniques based on geometry and introduce graph level self-supervised learning with geometric priors in both data and latent spaces in Chap. 4, and then see how we can enhance generalization performance for classification methods through automatic auxiliary task discovery in complex topological spaces (manifolds) in Chap. 5.

Chapter 4

Graph-level Representation Learning with Joint-Embedding Predictive Architectures

4.1 Introduction

As mentioned numerous times in this thesis, graph data is ubiquitous in the real world due to its ability to universally abstract various concepts and entities [166, 241]. To deal with this widespread data structure, Graph Neural Networks (GNNs) [211, 125, 76, 242] have established themselves as a staple solution. Nevertheless, most applications of GNNs usually rely on ground-truth labels for training. The growing amount of graph data in fields such as bioinformatics, chemoinformatics, and social networks makes manual labeling laborious, sparking significant interest in unsupervised graph representation learning.

A particularly emergent area in this line of research is self-supervised learning (SSL). In SSL, alternative forms of supervision are created stemming from the input signal. This process is then typically followed by invariance-based or generative-based pretraining [155, 7]. Invariance-based approaches optimize the model to produce comparable embeddings for different views of the input signal. A common paradigm associated with this procedure is contrastive learning [233]. Typically, these alternative views are created by a data augmentation procedure. The views are then passed through their respective encoder networks (which may share weights), as shown in Fig. 4.1a. Finally, an energy function, usually framed as a distance, acts on the latent embeddings. In the graph domain, several works have applied contrastive learning by designing graph-specific augmentations [269], using multi-view learning [94] and even adversarial learning [230]. Invariance-based pretraining

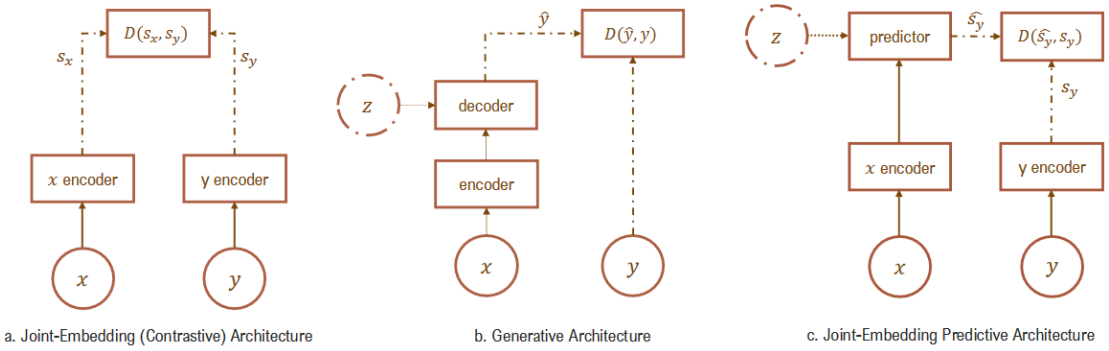


Fig. 4.1 Illustration of the SSL approaches discussed in this paper: (a) Joint-Embedding (Contrastive) Architectures learn to create similar embeddings for inputs x and y that are compatible with each other and dissimilar embeddings otherwise. This compatibility is implemented in practice by creating different views of the input data. (b) Generative Architectures reconstruct a signal y from an input signal x by conditioning the decoder network on additional (potentially latent) variables z. (c) Joint-Embedding Predictive Architectures act as a bridge: They utilize a predictor network that processes the context x and is conditioned on additional (potentially latent) variables to predict the embedding of the target y *in latent space*.

is effective but comes with several drawbacks i.e., the necessity to augment the data and process negative samples, which limits computational efficiency. In order to learn semantic embeddings that are useful for downstream tasks, the augmentations must also be non-trivial.

Generative-based pretraining methods, on the other hand, typically remove or corrupt portions of the input and predict them using an autoencoding procedure [245, 95] or rely on autoregressive modeling [30, 110]. Fig. 4.1b depicts the typical instantiation of these methods: The input signal x is fed into an encoder network that constructs the latent representation, and from it a decoder generates $\hat{y}$, the data corresponding to the target signal y. The energy function is then applied in data space, often through a reconstruction error. Generative models generally display strong overfitting tendencies [237] and can be non-trivial to train due to issues such as mode collapse [3]. Moreover, the features they learn are not always useful for downstream tasks [177]. An intuitive explanation of this problem is that generative models have to estimate a data distribution that is usually quite complex, so the latent representations must be directly descriptive of the whole data space [156]. This can become even more problematic for graphs because they live in a non-Euclidean and inhomogenous data space. Despite the aforementioned issues, masked autoencoding has recently shown promising results also in the graph domain with appropriately designed models [107, 231].

Inspired by the innovative Joint-Embedding Predictive Architecture (JEPA) [134, 7], we propose Graph-JEPA, a JEPA for the graph domain that can learn graph-level representations

by bridging contrastive and generative models. As illustrated in Fig. 4.1c, a JEPA has two encoder networks that receive the input signals and produce the corresponding representations. The two encoders can potentially be different models and don't need to share weights. A predictor module outputs a prediction of the latent representation of one signal based on the other, possibly conditioned on another variable. Graph-JEPA does not require any negative samples or complex data augmentation, and by operating in the latent space avoids the pitfalls associated with learning high-level details needed to fit the data distribution. However, the graph domain presents several challenges needed to properly design such an architecture: context and target extraction, designing a latent prediction task optimal for graph-level concepts, and learning expressive representations. In response to these questions, we equip Graph-JEPA with a specific masked modeling objective. The input graph is first divided into several subgraphs, and then the latent representation of randomly chosen target subgraphs is predicted given a context subgraph. The subgraph representations are consequently pooled to create a graph-level representation that can be used for downstream tasks.

The nature of graph-level concepts is often assumed to be hierarchical [268]. We conjecture that the typical latent reconstruction objective used in current JEPA formulations is not enough to provide optimal downstream performance. To this end, we design a prediction objective that starts by expressing the target subgraph encoding as a high-dimensional description of the hyperbolic angle. The predictor module is then tasked with predicting the location of the target in the 2D unit hyperbola. This prediction is compared with the target coordinates obtained by using the aforementioned hyperbolic angle. In this self-predictive setting, we explain why the stop-gradient operation and a simple predictor parametrization are useful to prevent representation collapse. To experimentally validate our approach, we evaluate Graph-JEPA against established contrastive and generative graph-level SSL methods across various graph datasets from different domains. Our proposed method demonstrates superior performance, outperforming most competitors while maintaining efficiency and ease of training. Notably, we observe from our experiments that Graph-JEPA can run up to 2.5x faster than Graph-MAE [107] and 8x faster than MVGRL [94]. Finally, we empirically demonstrate Graph-JEPA's ability to learn highly expressive graph representations by showing that a linear classifier trained on the learned representations almost perfectly distinguishes pairs of non-isomorphic graphs that the 1-WL test cannot differentiate.

4.2 Related work

4.2.1 Self-Supervised Graph Representation Learning

Graph Neural Networks (GNNs)[256, 211, 125, 242] are now established solutions to different graph machine learning problems such as node classification, link prediction, and graph classification. Nevertheless, the cost of labeling graph data is quite high, given the immense variability of graph types and the information they can represent. To alleviate this problem, SSL on graphs has become a research frontier, where we can distinguish between two major groups of methods[261, 155]:

Contrastive Methods. Contrastive learning methods usually minimize an energy function [102, 89] between different views of the same data. InfoGraph [229] maximizes the mutual information between the graph-level representation and the representations of substructures at different scales. GraphCL [269] works similarly to distance-based contrastive methods in the imaging domain. The authors first propose four types of graph augmentations and then perform contrastive learning based on them. The work of [94] goes one step further by contrasting structural views of graphs. They also show that a large number of views or multiscale training does not seem to be beneficial, contrary to the image domain. Another popular research direction for contrastive methods is learning graph augmentations and how to leverage them efficiently [230, 116]. Contrastive learning methods typically require a lot of memory due to data augmentation and negative samples. Graph-JEPA is much more efficient than typical formulations of these architectures, given that it does not require any augmentations or negative samples. Another major difference is that the prediction in latent space in JEPAs is done through a separate predictor network rather than using the common Siamese structure [28](Fig. 4.1a vs. 1c).

Generative Methods. The goal of generative models is to recover the data distribution, an objective that is typically implemented through a reconstruction process. In the graph domain, most generative architectures that are also used for SSL are extensions of Auto-Encoder (AE) models [103] architectures. These models learn an embedding from the input data and then use a reconstruction objective with (optional) regularization to learn the data distribution. Kipf and Welling [126] extended the framework of AEs and VAEs [124] to graphs by using a GNN as an encoder and the reconstruction of the adjacency matrix as the training objective. However, the results on downstream tasks with embeddings learned in this way are often unsatisfactory compared with contrastive learning methods, a tendency also observed in other domains [155]. A recent and promising direction is masked autoencoding (MAE)

[95], which has proved to be a very successful framework for the image and text domains. GraphMAE [107] is an instantiation of MAEs in the graph domain, where the node attributes are perturbed and then reconstructed, providing a paradigm shift from the structure learning objective of GAEs. S2GAE [231] is one of the latest GAEs, which focuses on reconstructing the topological structure but adds several auxiliary objectives and additional designs. Our architecture differs from generative models in that it learns to predict directly in the latent space, thereby bypassing the necessity of remembering and overfitting high-level details that help maximize the data evidence (Fig. 4.1b vs. 1c).

4.2.2 Joint-Embedding Predictive Architectures

Joint-Embedding Predictive Architectures [134] are a recently proposed design for SSL. The idea is similar to both generative and contrastive approaches, yet JEPAs are non-generative since they cannot directly predict y from x, as shown in Fig. 4.1c. The energy of a JEPA is given by the prediction error in the embedding space, not the input space. These models can intuitively be understood as a way to capture abstract dependencies between x and y, potentially given another latent variable z. It is worth noting that the different models comprising the architecture may differ in terms of structure and parameters. An in-depth explanation of Joint-Embedding Predictive Architectures and their connections to human representation learning is provided by LeCun [134]. Some works acknowledged that latent self-predictive architectures were effective [82, 43] even before JEPAs effectively became synonymous with this concept. Inspired by these trends, a number of related works have tried to employ latent prediction objectives for graph SSL, showing advantages mostly compared to contrastive learning. Thakoor et al. [232] perform latent self-prediction on augmented views of a graph in a similar fashion to BYOL [82], while Zhang et al. [277] rely on ideas from Canonical Correlation Analysis to frame a learning objective that preserves feature invariance and forces decorrelation when necessary. The work of Lee et al. [138] presents a model that learns latent positive examples through a k-NN and clustering procedure in the transductive setting, while Xie et al. [260] combine instance-level reconstruction (generative pretraining) and feature-level invariance (latent prediction). Given that these models learn using a latent self-predictive objective, similar to ours, we will refer to them also using the term self-predictive in the rest of the paper. Unlike previously proposed methods, Graph-JEPA operates exclusively in the latent space and implements a novel training task without the need for data augmentation. At the current state of the art, the JEPA framework has been implemented for images [7], video[14, 13], and audio [64]. We propose the first architecture implementing the modern JEPA principles for the graph domain and use it to learn graph-level representations.

4.3 Method

A general overview. We consider graphs G defined as $G = (V, E)$ where $V = \{v_1 \dots v_N\}$ is the set of nodes, with a cardinality $|V| = N$, and $E = \{e_1 \dots e_M\}$ is the set of edges, with a cardinality $|E| = M$. For simplicity of the exposition, we consider symmetric, unweighted graphs, although our method can be generalized to weighted or directed graphs. In this setting, G can be represented by an adjacency matrix $A \in \{0,1\}^{N \times N}$, with $A_{ij} = 1$ if nodes v_i and v_j are connected and 0 otherwise. Fig. 4.2 provides the overview of the proposed architecture. The high-level idea of Graph-JEPA is to divide the input graph into subgraphs (patches) [98] and then predict the representation of a randomly chosen target subgraph from the representation of a single context subgraph [7]. Again, we would like to stress that this masked modeling objective is realized in latent space without needing negative (or positive) samples. The subgraph representations are then pooled to create a vector representation for the whole graph, i.e., a graph-level representation. Therefore, the learning procedure consists of a sequence of operations: 1. Spatial Partitioning; 2. Subgraph Embedding; 3. Context and Target Encoding; 4. Latent Target Prediction.

4.3.1 Spatial Partitioning

We base the initial part of our architecture on the recent work of [98], but similar ideas consisting of graph partitioning have been proposed before for Graph SSL [115]. This step consists of creating different subgraphs (patches), similar to how Vision Transformers (ViT) [56] operate on images. We rely on the METIS [120] graph clustering algorithm, which partitions a graph into a pre-defined, non-overlapping number of clusters p (Fig. 4.2a), such that the number of within-cluster links is much higher than between-cluster links. Note that having non-overlapping subgraphs can be problematic since edges can be lost in this procedure, and it is possible to end up with empty "subgraphs". In order to maintain a simple notion of locality in each subgraph and avoid completely empty subgraphs, a one-hop neighborhood expansion[1] of the nodes in each extracted subgraph is performed (Fig. 4.2b).

4.3.2 Subgraph Embedding

After partitioning the graph, we learn a representation for each subgraph through a GNN (Fig. 4.2c.). The specific choice of the GNN is arbitrary and depends on what properties one wishes to induce in the representation. The learned node embeddings are mean pooled to create a

[1] Connecting all (not already present) adjacent nodes from the input graph with each node present in the subgraph.

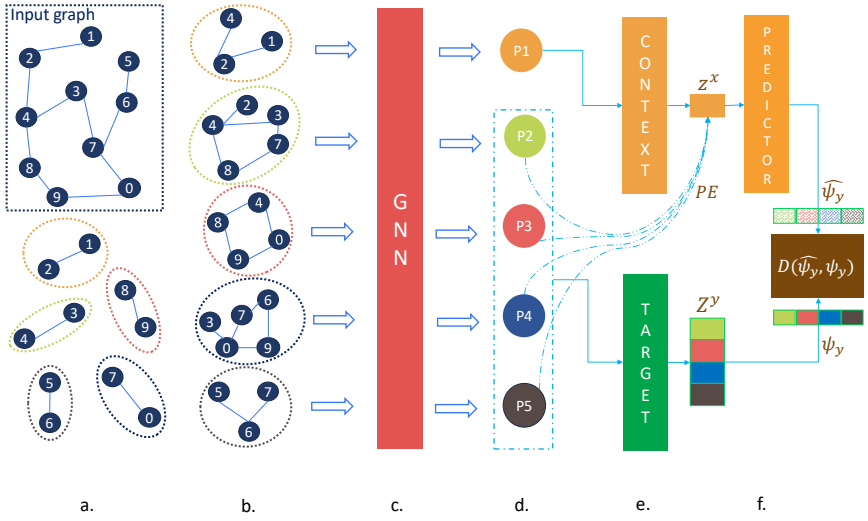


Fig. 4.2 A complete overview of Graph-JEPA. We first extract non-overlapping subgraphs (patches) (a.), perform a 1-hop neighborhood expansion (b.), and encode the subgraphs with a GNN (c.). After the subgraph encoding, one is randomly picked as the context and m others as the targets (d.), and they are fed into their respective encoders (e.). The embeddings generated from the target encoder are used to produce the target subgraphs' coordinates ψ_y. Finally, the predictor network is tasked with directly predicting the coordinates $\hat{\psi}_y$ for each target subgraph based on the context embedding and the positional embedding of each target subgraph (f.). A regression loss acts as the energy function D between the predicted and target coordinates. Note that the extracted subgraphs in (a.) and (b.) are meant for illustrative purposes only. The number of nodes in each subgraph can vary.

vector representation for each subgraph: $\{h_1 ... h_p\}, h \in \mathbb{R}^d$. Given that these embeddings will be used as context or target variables, providing additional information regarding the subgraphs is key in order to guide the predictor network. Otherwise, the prediction task might be too difficult. Thus, we propose to use a positional embedding for each subgraph, which is implemented as the maximum Random Walk Structural Embedding (RWSE) of all the nodes in that subgraph. In this way, the position is characterized consistently for each patch. Formally, a RWSE [59] for a node v can be defined as:

$$P_v = (M_{ii}, M_{ii}^2, \ldots, M_{ii}^k) \tag{4.1}$$

with $P_v \in \mathbb{R}^k$, $M^k = (D^{-1}A)^k$ the random-walk transition matrix of order k, and i is the index of node v in the adjacency matrix. Therefore, M_{ii}^k encodes the probability of node v landing to itself after a k-step random walk. Given a subgraph l, let V_l denote the set of nodes in l.

We define the subgraph RWSE as:

$$P_l = \max_{v \in V_l} P_v \tag{4.2}$$

where the max operation is performed elementwise. A straightforward explanation of the above definition is that the positional information of each subgraph is essentially contained in the node with the highest degree[2], which will act as an anchor reference when predicting the target subgraphs' latent representations. Intuitively, knowing P_l is useful for the prediction task because the features present in the most representative node will have been diffused the most (via the GNN) into the subgraph.

4.3.3 Context and Target Encoding

Given the subgraph representations and their respective positional embeddings, we frame the Graph-JEPA prediction task in a similar manner to I-JEPA [7]. The goal of the network is to predict the latent embeddings of randomly chosen target subgraphs, given one random context subgraph. The prediction is conditioned on positional information regarding each subgraph. At each training step, we choose one random subgraph as the context x and m others as targets $Y = \{y_1, \ldots, y_m\}$ (Fig. 4.2d). These subgraphs are processed by the context and target encoders (Fig. 4.2e) which are parametrized by Transformer encoder blocks (without self-attention for the context) where normalization is applied at first [262]. The target encoder uses Hadamard self-attention [98], but other choices, such as the standard self-attention mechanism [240] are perfectly viable. We can summarize this step as:

$$z^x = E_c(x),\ Z^y = E_t(Y), \tag{4.3}$$

with $z^x \in \mathbb{R}^d$ and $Z^y \in \mathbb{R}^{m \times d}$. At this stage, we can use the predictor network to directly predict Z^y from z^x. This is the typical formulation of JEPAs, also followed by the popular work of Assran et al. [7]. We argue that learning how to organize concepts for abstract objects such as graphs or networks directly in Euclidean space is suboptimal. In the following subsection, we propose a simple trick to bypass this problem using the encoding and prediction mechanisms in Graph-JEPA. A discussion in Section 4.4.3 will provide additional insights regarding this topic.

[2] A more suitable term would be the in-degree, but there is no difference in the undirected case.

4.3.4 Latent Target Prediction

Learning hierarchically consistent concepts [52] is considered a crucial aspect of human learning, especially during infancy and young age [207]. Networks in the real world often conform to some concept of hierarchy [185], and this assumption is frequently used when learning graph-level representations [268]. Thus, we conjecture that Graph-JEPA should operate in a hyperbolic space, where learned embeddings implicitly organize hierarchical concepts[192**?**]. This gives rise to another issue: commonly used hyperbolic (Poincaré) embeddings are known to have several tradeoffs between dimensionality and performance[208, 87], which severely limits the expressive ability of the model. Given that graphs can describe very abstract concepts, high expressivity in terms of model parameters is preferred. In simple words, we would ideally like to have a high-dimensional latent code that has a concept of hyperbolicity built into it.

To achieve this, we think of the target embedding as a high-dimensional representation of the hyperbolic angle, which allows us to describe each target patch through its position in the 2D unit hyperbola. Formally, given a target patch l, its embedding Z_l^y and positional encoding P_l, we express the latent target as:

$$\psi_l^y = \begin{pmatrix} cosh(\alpha_l^y) \\ \\ sinh(\alpha_l^y) \end{pmatrix}, \; \alpha_l^y = \frac{1}{N}\sum_{n=1}^{d} Z_l^{y(n)}, \tag{4.4}$$

where $cosh(\cdot)$ and $sinh(\cdot)$ are the hyperbolic cosine and sine functions. The predictor module is then tasked with directly locating the target in the unit hyperbola, given the context embedding and the target patch's positional encoding:

$$\hat{\psi}_l^y = W_2(\sigma(W_1(z^x + P_l) + b_1)) + b_2, \tag{4.5}$$

where W_n and b_n represent the n-th weight matrix and bias vector (i.e., n-th fully connected layer), σ is an elementwise non-linear activation function, and $\hat{\psi}_l^y \in \mathbb{R}^2$. This allows us to frame the learning procedure as a regression problem, and the whole network can be trained end-to-end (Fig. 4.2f). In practice, we use the smooth L1 loss as the distance function, as it is less sensitive to outliers compared to the typical L2 loss [77]:

$$L(y,\hat{y}) = \frac{1}{N}\sum_{n=1}^{N} s_n, \quad s_n = \begin{cases} 0.5(y_n - \hat{y}_n)^2/\beta, & \text{if } |y-\hat{y}| < \beta \\ |y-\hat{y}| - 0.5\beta, & \text{otherwise} \end{cases} \tag{4.6}$$

Thus, we are effectively measuring how far away the context and target patches are in the unit hyperbola of the plane, but the targets are actually described through a high dimensional latent code (Eq. 4.4). We explicitly show the differences between this choice and using the Euclidean or Hyperbolic distances as energy functions (in the latent space) in Section 4.4.3. Our proposed pretraining objective forces the context encoder to understand the differences in the hyperbolic angle between the target patches, which can be thought of as establishing an implicit hierarchy between them.

Preventing representation collapse. JEPAs are based on a self-distillation procedure. Therefore, they are by definition susceptible to representation collapse [7]. This is due to the nature of the learning process, where both the context and target representations have to be learned. We formalize this intuition with an example and argue why there is a need to adopt two well-known training tricks that are prevalent in the literature to prevent representation collapse: i) The stop-gradient operation on the target encoder followed by a momentum update (using an Exponential Moving Average (EMA) of the context encoder weights) [82, 43]; ii) a simpler parametrization of the predictor compared to the context and target networks (in terms of parameters)[41, 9];

Let us simplify the problem through the following assumptions: i) The predictor network is linear; ii) There is a one-to-one correspondence between context and target patches. (This holds also in practice due to Eq. 4.5); iii) The self-predictive task is a least-squares problem in a finite-dimensional vector space. Based on these assumptions, we can rewrite the context features as $X \in \mathbb{R}^{n\times d}$, the target coordinates as $Y \in \mathbb{R}^{n\times 2}$, and the weights of the linear model as $W \in \mathbb{R}^{d\times 2}$. The optimal weights of the predictor are given by solving:

$$\arg\min_{W} \|XW - Y\|^2 \tag{4.7}$$

where $\|.\|$ indicates the Frobenius norm. The (multivariate) OLS estimator can give the solution to this problem by setting W to:

$$W = (X^T X)^{-1} X^T Y \tag{4.8}$$

Plugging Eq. 4.8 into Eq. 4.7 and factorizing Y, the least squares solution leads to the error:

$$\left\|(X(X^T X)^{-1} X^T - I_n)Y\right\|^2 \tag{4.9}$$

Thus, the optimality of a linear predictor is defined by the orthogonal projection of Y onto the orthogonal complement of a subspace of $Col(X)$. As is commonly understood, this translates

to finding the linear combination of X that is closest, in terms of $\|\cdot\|^2$, to Y. Similarly to what was shown by Richemond et al. [205], we argue that this behavior unveils a key intuition: The target encoder, *which estimates Y* must not share weights or be optimized via the same optimizer as the context encoder. If that were the case, the easiest solution to Eq. 4.9 would be learning a representation that is orthogonal to itself, i.e., the $\mathbf{0}$ vector, leading to representation collapse. Using a well-parametrized EMA update is what allows us to bypass this problem. In practice, even with the slower dynamics induced by the EMA procedure, it is possible to immediately encounter a degenerate solution with a non-linear and highly expressive network. For instance, consider a scenario where the target subgraphs are straightforward and similar. In this case, if the predictor network is sufficiently powerful, it can predict the target representations even without a well-learned context representation. Since the target encoder weights are updated via the EMA procedure, the learned representations will tend to be uninformative. Therefore, implementing the predictor network as a simpler and less expressive network is crucial to achieving the desired training dynamics.

4.4 Experiments

The experimental section introduces the empirical evaluation of the Graph-JEPA model in terms of downstream performance on different graph datasets and tasks, along with additional studies on the latent space's structure and the encoders' parametrization. Furthermore, a series of ablation studies are presented in order to elucidate the design choices behind Graph-JEPA.

4.4.1 Experimental setting

We use the TUD datasets [183] as commonly done for graph-level SSL [230, 231]. We utilize seven different graph-classification datasets: PROTEINS, MUTAG, DD, REDDIT-BINARY, REDDIT-MULTI-5K, IMDB-BINARY, and IMDB-MULTI. We report the accuracy of ten-fold cross-validation for all classification experiments over five runs (with different seeds). It is worth noting that we retrain the Graph-JEPA model for each fold without ever having access to the testing partition in both the pretraining and fine-tuning stages. We use the ZINC dataset for graph regression and report the Mean Squared Error (MSE) over ten runs (with different seeds), given that the testing partition is already separated. To produce the unique graph-level representations, we feed all the subgraphs through the trained target encoder and then use mean pooling, obtaining a single feature vector $z_G \in \mathbb{R}^d$ that represents the whole graph. This high-dimensional is then used to fit a *linear model* with L2 regularization for the

Table 4.1 Values of Graph-JEPA specific hyperparameters for the experiments on the TUD datasets.

Hyperparameter	PROTEINS	MUTAG	DD	REDDIT-B	REDDIT-M5	IMDB-B	IMDB-M	ZINC
# Subgraphs	32	32	32	128	128	32	32	32
# GNN Layers	2	2	3	2	2	2	2	2
# Encoder Blocks	4	4	4	4	4	4	4	4
Embedding size	512	512	512	512	512	512	512	512
RWSE size	20	15	30	40	40	15	15	20
# context - # target	1 - 2	1 - 3	1 - 4	1 - 4	1 - 4	1- 4	1- 4	1- 4

downstream task. Specifically, we employ Logistic Regression with L2 regularization on the classification datasets and Ridge Regression for the ZINC dataset. For the datasets that do not natively have node and edge features, we use a simple constant (0) initialization. The subgraph embedding GNN (Figure 4.2c.) consists of the GIN operator with support for edge features [109], often referred to as GINE. The neural network modules were trained using the Adam optimizer [123] and implemented using PyTorch [199] and PyTorch-Geometric [65], while the linear classifiers and cross-validation procedure were implemented through the Scikit-Learn library [201]. All experiments were performed on Nvidia RTX 3090 GPUs. Finally, 4.1 shows the JEPA-specific hyperparameters used for the following experiments.

4.4.2 Downstream performance

For the experiments on downstream performance, we follow Suresh et al. [230] and also report the results of a fully supervised Graph Isomorphism Network (GIN) [263], denoted F-GIN in Table 4.2. We compare Graph-JEPA against four contrastive methods, two generative methods, and one latent self-predictive method [260] (which also regularizes through instance-level reconstruction). As can be seen in Table 4.2, Graph-JEPA achieves competitive results on all datasets, setting the state-of-the-art as a pretrained backbone on five different datasets and coming second on one (out of eight total). Overall, our proposed framework learns semantic embeddings that work well on different graphs, showing that Graph-JEPA can be utilized as a general pretraining method for graph-level SSL. Notably, Graph-JEPA works well for both classification and regression and performs better than a supervised GIN on all classification datasets. We also provide results with BGRL [232], a node-level latent self-predictive strategy. We train this model using the official code and hyperparameters and then mean-pool the node representations for the downstream task. The results are underwhelming compared to the models reporting graph-level performance, which is to be expected considering that methods that also perform well on graph-level learning are appropriately designed.

Table 4.2 Performance of different graph SSL techniques on various TUD benchmark datasets, ordered by pretraining type: contrastive, generative, and self-predictive. F-GIN is an end-to-end supervised GIN and serves as a reference for the performance values. The results of the competitors are taken as the best values from [94, 230, 231]. "-" indicates missing values from the literature. The **best results** are reported in boldface, and the second best are underlined. For the sake of completeness, we also report the training and evaluation results of GraphMAE on the DD, REDDIT-M5, and ZINC datasets in *italics*, along with the results of a node-level self-predictive method (BGRL), which does not originally report results on graph-level tasks.

Model	PROTEINS ↑	MUTAG ↑	DD ↑	REDDIT-B ↑	REDDIT-M5 ↑	IMDB-B ↑	IMDB-M ↑	ZINC ↓
F-GIN	72.39 ± 2.76	90.41 ± 4.61	74.87 ± 3.56	86.79 ± 2.04	53.28 ± 3.17	71.83 ± 1.93	48.46 ± 2.31	0.254 ± 0.005
InfoGraph [229]	72.57 ± 0.65	87.71 ± 1.77	75.23 ± 0.39	78.79 ± 2.14	51.11 ± 0.55	71.11 ± 0.88	48.66 ± 0.67	0.890 ± 0.017
GraphCL [269]	72.86 ± 1.01	88.29 ± 1.31	74.70 ± 0.70	82.63 ± 0.99	53.05 ± 0.40	70.80 ± 0.77	48.49 ± 0.63	0.627 ± 0.013
MVGRL [94]	-	-	-	84.5 ± 0.6	-	74.2 ± 0.7	51.2 ± 0.5	-
AD-GCL-FIX [230]	73.59 ± 0.65	89.25 ± 1.45	74.49 ± 0.52	85.52 ± 0.79	53.00 ± 0.82	71.57 ± 1.01	49.04 ± 0.53	0.578 ± 0.012
AD-GCL-OPT [230]	73.81 ± 0.46	89.70 ± 1.03	75.10 ± 0.39	85.52 ± 0.79	54.93 ± 0.43	72.33 ± 0.56	49.89 ± 0.66	0.544 ± 0.004
GraphMAE [107]	75.30 ± 0.39	88.19 ± 1.26	*74.27 ± 1.07*	88.01 ± 0.19	*46.06 ± 3.44*	75.52 ± 0.66	51.63 ± 0.52	*0.935 ± 0.034*
S2GAE [231]	**76.37 ± 0.43**	88.26 ± 0.76	-	87.83 ± 0.27	-	**75.76 ± 0.62**	**51.79 ± 0.36**	-
BGRL [232]	70.99 ± 3.86	74.99 ± 8.83	71.52 ± 2.97	50 ± 0	20 ± 0.1	0.5 ± 0	0.33 ± 0	1.2 ± 0.011
LaGraph [260]	75.2 ± 0.4	90.2 ± 1.1	78.1 ± 0.4	90.4 ± 0.8	56.4 ± 0.4	73.7 ± 0.9	-	-
Graph-JEPA	75.67 ± 3.78	**91.25 ± 5.75**	**78.64 ± 2.35**	**91.99 ± 1.59**	**56.73 ± 1.96**	73.68 ± 3.24	50.69 ± 2.91	**0.434 ± 0.014**

We further explore the performance of our model on the synthetic EXP dataset [1], compared to end-to-end supervised models. This experiment aims to empirically verify if Graph-JEPA can learn highly expressive graph representations (in terms of the commonly used WL hierarchy [184]) without relying on supervision. The results in Table 4.3 show that our model is able to perform much better than commonly used GNNs. Given its local and global exchange of information, this result is to be expected. Most importantly, Graph-JEPA almost matches the flawless performance achieved by He et al. [98], who train fully supervised.

4.4.3 Exploring the Graph-JEPA latent space

As discussed in Section 4.3.4, the choice of energy function greatly impacts the learned representations. Given the latent prediction task of Graph-JEPA, we expect the latent representations to display hyperbolicity. The predictor network approximates the location on the unit hyperbola to best match the generated target coordinates (Eq. 4.6). Thus, the network is actually trying to estimate a space that can be considered a particular section of the hyperboloid model [204], where hyperbolas appear as geodesics. We are, therefore, evaluating our energy in a restricted part of hyperbolic space. As mentioned before, we find this task to offer great flexibility as it is straightforward to implement and it is computationally efficient compared to the hyperbolic distance used to typically learn hyperbolic embeddings in the Poincaré ball model [192]. Table 4.4 provides empirical evidence for our conjectures

Table 4.3 Classification accuracy on the synthetic EXP dataset [1], which contains 600 pairs of non-isomorphic graphs that are indistinguishable by the 1-WL test. Note that the competitor models are all trained with *end-to-end supervision*. The **best result** is reported in boldface, and the second best is underlined. Performances for all competitor models are taken from [98].

Model	Accuracy ↑
GCN [125]	51.90 ± 1.96
GatedGCN [26]	51.73 ± 1.65
GINE [263]	50.69 ± 1.39
GraphTransformer [58]	52.35 ± 2.32
Graph-MLP-Mixer [98]	**100.00 ± 0.00**
Graph-JEPA	98.77 ± 0.99

Table 4.4 Comparison of Graph-JEPA performance for different distance functions. The optimization for Poincaré embeddings in higher dimensions is problematic, as shown by the *NaN* loss on the IMDB-B dataset. LD stands for Lower Dimension, where we use a smaller embedding size.

Distance function	Ours	Euclidean	Hyperbolic	Euclidean (LD)	Hyperbolic (LD)
MUTAG	**91.25 ± 5.75**	87.04 ± 6.01	89.43 +- 5.67	86.63 ± 5.9	86.32 ± 5.52
REDDIT-M	**56.73 ± 1.96**	56.55 ± 1.94	56.19 +- 1.95	54.84 ± 1.6	55.07 ± 1.83
IMDB-B	73.68 ± 3.24	**73.76 ± 3.46**	*NaN*	72.5 ± 3.97	73.4 ± 4.07
ZINC	**0.434 ± 0.01**	0.471 ± 0.01	0.605 +- 0.01	0.952 ± 0.05	0.912 ± 0.04

regarding the suboptimality of Euclidean or Poincaré embeddings on 4 out of the 8 datasets initially presented in Table 4.2, where we make sure to choose different graph types for a fair comparison. The results reveal that learning the distance between patches in the 2D unit hyperbola provides a simple way to get the advantages of both embedding types. Hyperbolic embeddings must be learned in lower dimensions due to stability issues [271], while Euclidean ones do not properly reflect the dependencies between subgraphs and the hierarchical nature of graph-level concepts. Our results suggest that the hyperbolic (Poincaré) distance is generally a better choice than the Euclidean distance in lower dimensions, but it is computationally unstable and expensive in high dimensions. The proposed approach provides the best overall results. We provide a qualitative example of how the embedding space is altered from our latent prediction objective in Fig. 4.3.

4.4.4 Additional insights and ablation studies

Model efficiency. In an attempt to characterize the efficiency of our proposed model, we perform a simple experimental check. In Table 4.5, we compare the total training time needed

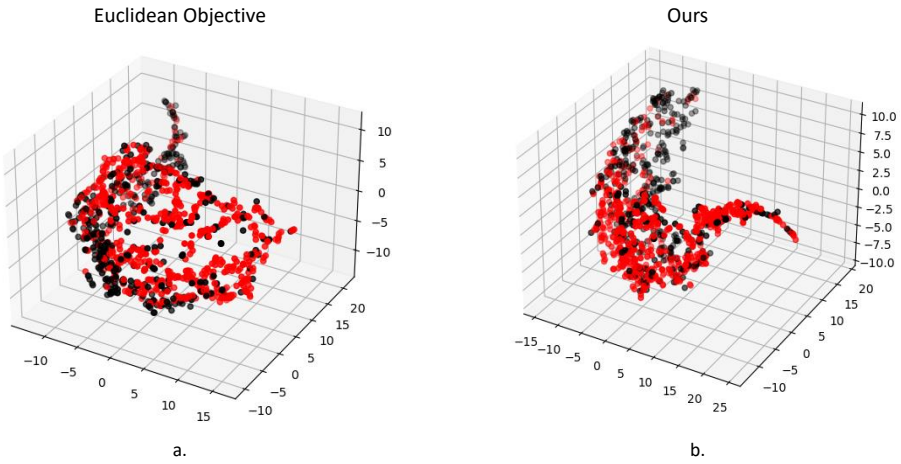


Fig. 4.3 3D t-SNE[238] of the latent representations used to train the linear classifier on the DD dataset. The change in the curvature of the embedding using the Graph-JEPA objective (b.) is noticeable. Best viewed in color.

Table 4.5 Total training time and model parameters of MVGRL, GraphMAE, and Graph-JEPA for pretraining (single run) based on the optimal configuration for downstream performance. OOM stands for Out-Of-Memory.

Dataset	Model	Num. parameters	Training time
IMDB	MVGRL	3674118	$\sim$ 7 min
	GraphMAE	2257193	$\sim$ 1.5 min (1min 36sec)
	Graph-JEPA	**19219460**	$<$ **1min (56 sec)**
REDDIT-M5	MVGRL	4198406	*OOM*
	GraphMAE	2784555	$\sim$ 46 min
	Graph-JEPA	**19245060**	$\sim$ **18 min**

Table 4.6 Performance when parametrizing the context and target encoders through MLPs vs using the proposed Transformer encoders.

Dataset	Transformer Encoders	MLP Encoders
MUTAG	**91.25 ± 5.75**	90.5 ± 5.99
REDDIT-M5	**56.73 ± 1.96**	56.21 ± 2.29
IMDB-B	73.68 ± 3.24	**74.26 ± 3.56**
ZINC	**0.434 ± 0.01**	0.472 ± 0.01

for different Graph-SSL strategies to provide a representation that performs optimally on the downstream task. We show results on the datasets with the largest graphs from Table 4.2: IMDB and REDDIT-M5. While the runtime is hardware-dependent, all experiments are performed on the same machine. Graph-JEPA displays superior efficiency and promising scaling behavior. The presented runtime is naturally dependent on the self-supervised scheme used in each model, so we do not regard it as a definitive descriptor, but rather an indicator of the potential of fully latent self-predictive models.

MLP parametrization. Table 4.6 contains the results of parametrizing the whole architecture, other than the initial GNN encoder, through MLPs. This translates to not using the Attention mechanism at all. For this experiment, and also the following ablations, we consider 4 out of the 8 datasets initially presented in Table 4.2, making sure to choose different graph domains for a fair comparison. Graph-JEPA still manages to perform well, showing the flexibility of our architecture, even though using the complete Transformer encoders leads to better overall performance and less variance in the predictions.

Positional embedding. Following [98], it is possible to use the RWSE of the patches as conditioning information. Formally, let $B \in \{0,1\}^{p \times N}$ be the patch assignment matrix, such that $B_{ij} = 1$ if $v_j \in p_i$. We can calculate a coarse patch adjacency matrix $A' = BB^T \in \mathbb{R}^{p \times p}$, where each A'_{ij} contains the node overlap between p_i and p_j. The positional embedding can then be calculated for each patch by simply using the RWSE described in Eq. 4.1 on A'. We test Graph-JEPA with these relative positional embeddings and find that they still provide good performance but consistently fall behind the node-level (global) RWSE that we employ in our formulation (Table 4.7b). An issue of these relative patch RWSEs is that the number of shared neighbors can obscure the local peculiarities of each patch, rendering the context given to the predictor more ambiguous.

Random subgraphs. A natural question that arises in our framework is how to design the spatial partitioning procedure. Using a structured approach like METIS [120] is intuitive

Table 4.7 (a) Performance when using node-level vs patch-level RWSEs. (b) Performance when extracting subgraphs with METIS vs. using random subgraphs.

(a)

Dataset	METIS	Random
MUTAG	91.25 ± 5.75	**91.58 ± 5.82**
REDDIT-M5	**56.73 ± 1.96**	56.08 ± 1.69
IMDB-B	**73.68 ± 3.24**	73.52 ± 3.08
ZINC	0.434 ± 0.01	**0.43 ± 0.01**

(b)

Dataset	Node RWSE	Patch RWSE
MUTAG	**91.25 ± 5.75**	91.23 ± 5.86
REDDIT-M5	**56.73 ± 1.96**	56.01 ± 2.1
IMDB-B	**73.68 ± 3.24**	73.58 ± 4.47
ZINC	**0.434 ± 0.01**	0.505 ± 0.005

and leads to favorable results. Another option would be to extract random, non-empty subgraphs as context and targets. As seen in Table 4.7a, the random patches also provide strong performance, showing that the proposed JEPA architecture is not reliant on the initial input, as is the case for many methods that rely on data augmentation for view generation [138]. Even though our results show that using a structured way to extract the patches might not be necessary, random sampling can be problematic for larger graphs. Thus, we advocate extracting subgraphs with METIS as it is a safer option in terms of generalizability across different graphs and the inductive bias it provides.

4.5 Conclusion

In this work, we introduce a new Joint Embedding Predictive Architecture (JEPA) [134], for graph-level Self-Supervised Learning (SSL). An appropriate design of the model, both in terms of data preparation and pretraining objective, reveals that it is possible for a neural network to self-organize the semantic knowledge embedded in a graph, demonstrating competitive performance in different graph data and tasks. Future research directions include extending the proposed method to nodes and edge-level learning, theoretically exploring the expressiveness of Graph-JEPA, and gaining more insights into the optimal geometry of the embedding space for graph SSL.

Chapter 5

Data-driven Auxiliary Learning via Latent Geometric Disentanglement

5.1 Introduction

Human learning is often considered to be a combination of processes (e.g., high-level acquired skills and evolutionary encoded physical perception) that are used together and can be transferred from one problem to another. Inspired by this, *Multi-Task Learning (MTL)* [36] represents a machine learning paradigm where multiple tasks are learned together to improve the generalization ability of a model by using shared knowledge that derives from considering different aspects of the input. Specifically, this is achieved by jointly optimizing the model's parameters across different tasks, allowing the model to learn task-specific and task-shared representations simultaneously. As a result, MTL can lead to better generalization, improved efficiency at inference time, and enhanced performance on individual tasks by exploiting their underlying relationships.

A specific form of this learning approach, referred to as *auxiliary learning*, has garnered considerable interest in recent years [149]. In particular, auxiliary learning is a specific type of MTL, where auxiliary tasks are simply tasks that operate on the same (or different) input data as the main one, intentionally crafted to learn an effective, shared representation able to boost the performance on the principal task. At the state-of-the-art, auxiliary tasks are found by meta-learning [153, 188], but this requires the a priori definition of the hierarchy of the desired auxiliary tasks and is computationally inefficient. Thus, the question is: can we discover with no prior knowledge one or more additional auxiliary tasks to improve the performance of the principal task? In this work, we explore this difficult problem by proposing *Detaux*, a weakly supervised strategy that discovers auxiliary classification tasks

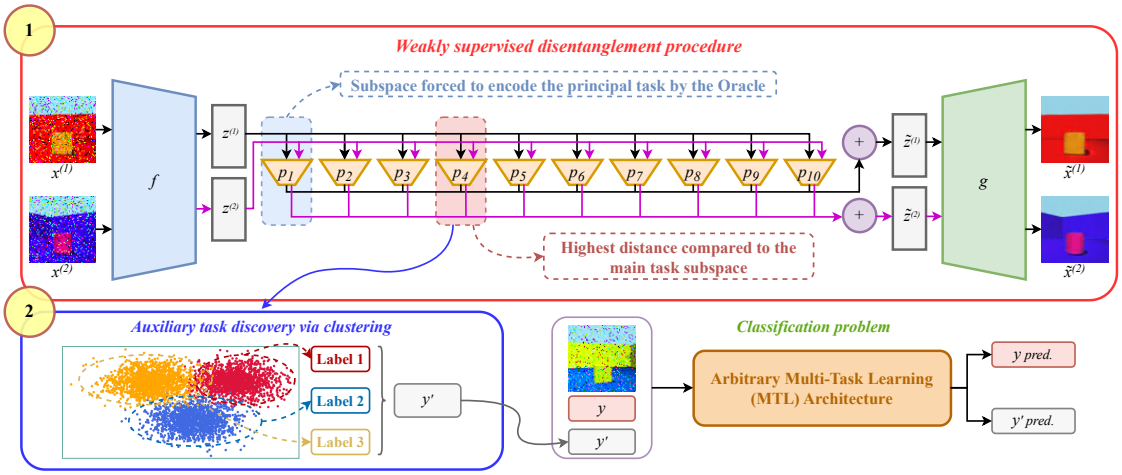


Fig. 5.1 *Detaux* involves two steps: 1) First, we use weakly supervised disentanglement to isolate the structural features specific to the principal task in one subspace (red rectangle at the top of the image). 2) Next, we identify the subspace with the most disentangled factor of variation related to the principal task, and through a clustering module, we obtain new labels (blue rectangle in the bottom left part of the image). These can be used to create a new classification task that can be combined with the principal task in any MTL model (bottom right part of the image).

that enable solving a single-task classification problem in a multi-task fashion. Specifically, *Detaux* is capable of individuating unrelated auxiliary tasks: unrelatedness in MTL means to have two or more tasks that do not share any features, as proven to be effective in the MTL literature [249, 282, 197, 114, 281, 151]. As a result of this automated procedure, new auxiliary classification tasks can be discovered on a given dataset, thus opening the possibility of MTL. In this work, we focus specifically on unveiling a single auxiliary task.

In particular, our method takes roots in the idea from [197], which starts with two groups of tasks, the principal task and the auxiliary tasks, which are given and known to be unrelated, and assumes the claim that joint learning of unrelated tasks can improve the performance on the principal task. They propose to generate a shared low-dimensional representation for both the principal task and the unrelated auxiliary tasks forcing these two representations to be orthogonal.

The procedure from [197] exploits a linear classifier and requires the knowledge of the labels for both the principal task and the auxiliary tasks. With our method, we aim to follow a similar process giving up on the supervision and fostering non-linear classifiers estimated by neural networks. The proposed method generates auxiliary tasks in such a way that their labels implicitly drive an MTL network to understand the unrelatedness between the tasks. Our idea is to work in a specific representation space, a product manifold, to unveil the auxiliary

tasks for a given principal task. We get inspiration from [70], who discovered the product manifold as a convenient representation basis for disentanglement. In particular, as depicted in Fig. 5.1, we extract task-specific features using weakly supervised disentanglement; then, we identify the most disentangled factor of variation within a subspace, and finally, we generate new labels via a clustering module to enable seamless integration with the primary task in any MTL model. Notably, this makes the proposed pipeline agnostic to the choice of the MTL model, given that the latter acts directly on the primary and generated auxiliary labels, as depicted in the bottom right of Figure 5.1. In this way, any MTL model can be chosen depending on several factors besides performance, such as efficiency, scalability, and resource constraints. In the experimental section, we utilize four different MTL models with *Detaux*, revealing its flexibility. This makes our proposed pipeline agnostic to the choice of the MTL model, given that the latter acts directly on the primary and auxiliary labels. In this way, any MTL model can be chosen, depending on several factors besides performance, such as efficiency, scalability, and resource constraints. In the experimental section, we utilize three different MTL models with *Detaux*, revealing its flexibility.

5.2 Related Work

We organize this section into three different parts, each one providing an overview of a topic related to our work: *i)* MTL and auxiliary learning; *ii)* disentanglement; and *iii)* existing studies on the relationship between MTL and disentanglement.

5.2.1 MTL and auxiliary learning

MTL, i.e., the procedure through which we can solve multiple learning problems at the same time [36], can help us reduce inference time, reach improved accuracy, and increase data efficiency [228]. When the adopted dataset contains annotation for multiple tasks, the challenges to face concern which tasks may work well together [273, 228, 67] or how to weigh the losses of different tasks [121] to create a better joint optimization objective. Numerous methods have recently emerged addressing the simultaneous resolution of multiple tasks [72, 239].

A different problem arises when we would like to use one of these methods but only one task is approachable, given the annotations in the considered dataset. Auxiliary task learning aims at maximizing the prediction performance on a principal task by supervising the model to additionally learn other tasks, as shown in [153, 188]. Therefore, auxiliary tasks are tasks of minor interest, or even irrelevant compared to the principal task we want to solve, and

thus can be seen as regularizers if learned simultaneously with the task of interest [149]. For example, [197] suggests that using two unrelated groups of tasks, where one of them is hosting the principal task, can lead to better performance, where unrelated means that the two groups of tasks are defined by an orthogonal set of features. Also in [149] the authors make use of seemingly unrelated tasks to help the learning on one principal task, this time without imposing any constraint on the feature structure. In our case, we are working in product manifold, which has been already shown by [70] as effective for separating embedding subspaces that are orthogonal by design.

Moreover, recent emerging techniques leverage meta-learning to effectively select the most appropriate auxiliary tasks or even autonomously create novel ones. [153] and [148] both train two neural networks simultaneously, a label-generation network to predict the auxiliary labels and a multi-task network to train the primary task alongside the auxiliary task. These, in contrast with our approach, require the a priori definition of a hierarchy binding the auxiliary labels to the principal task labels and present conflicting ideas on the possible semantic interpretation of the generated labels. Furthermore, they are computationally inefficient: meta-learning is a resource-intensive technique and requires the retraining of the entire architecture to change the employed multi-task method.

Even more recently, [53] propose to deconstruct existing objectives for NLP within a unified taxonomy, identifying connections between them, and generating new ones by selecting the best combinations from a cartesian product of the available options. Furthermore, [186] also used meta-learning, presenting a novel framework for generating new auxiliary objectives to address the niche problem of few-shot semi-supervised tabular learning. To the best of our knowledge, we are not aware of any other method that proposes a systematic approach for generating new labels from a disentangled latent space, in order to enable MTL classification when only the annotations for one task are given in the considered dataset.

5.2.2 Learning disentangled representations

Representing data in a space where different components are independent is a long-standing research topic in machine learning. The rise of deep learning in recent years led to proposed learning disentangled representations as an important aspect of unsupervised deep learning [22].

Recent literature has proposed several characterizations of disentanglement, whether that is in terms of group theory [99], metric and product spaces [70], or permutations of element-wise, nonlinear functions [106]. The seminal paper of [100] demonstrated that variational auto-encoders could learn to disentangle by enforcing the ELBO objective, while [42] relies on GANs and an information-theoretic view of disentanglement. Later works, such

as [60, 221, 193] extensively explored different directions and use cases. Work by [157] showed that completely unsupervised disentanglement was not possible due to the inability of the models to identify factors of variation. Soon after, the authors proposed weak supervision and having access to few labels as a way to bypass this limitation [158, 160].

In *Detaux*, we place ourselves in the same setting of [70], but control and force the disentanglement by supervision only on the known (principal) task. We describe in detail the main differences between the original method and our custom implementation in Section 5.4.

5.2.3 Relationship between MTL and disentanglement

[178] report a connection between disentangled representations and MTL, showing that disentangled features can improve the performance of multi-task networks, especially on data with previously unseen properties. Disentanglement is obtained by adversarial learning, forcing the encoded features to be minimally informative about irrelevant tasks. In this case, the tasks to be disentangled are known a priori, while in our case, only the principal task is known.

[266] propose a novel concept called "Knowledge Factorization". Exploiting the knowledge contained in a pre-trained multi-task network (called teacher), the idea is to train disentangled single-task networks (called students) to reduce the computational effort required by the final single-task network. The factorization of the teacher knowledge is dual: they provide structural factorization and representation factorization. In structural factorization, they split the net into a common-knowledge network and a task-specific network, based on mutual information.

Finally, [176] propose a disentanglement analysis of MTL models by creating a semi-synthetic dataset based on latent information in simple datasets. The authors run the latent information through randomly initialized fully-connected layers to create tasks that are harder than just recovering the simple factors. A CNN is then trained to produce a representation that fits these auxiliary tasks. The reported results may be seen as inconclusive, as they do not provide a clear indication of how disentangled representations directly impact MTL performance.

In our proposal, we show that disentanglement in a representation space can be used as a general prior for MTL, i.e., by using disentanglement to mine for auxiliary tasks, an MTL model extracts a model-specific embedding which exploits the combination of the principal and the newly discovered labels, thus improving the performance on the principal task.

5.3 Mathematical Background

To understand the core concepts of our research, let us delve into the mathematical background presented in [70]. Due to lack of space, we only provide a compact overview of the key concepts of this idea and refer the readers to the original paper for more details regarding the loss formulations and training process.

Given as input a collection of data, such as a set of labeled images, a disentanglement procedure should output a representation of these data, separating the different generative factors that produce all the variations observed in the data. The method proposed by [70] is based on the manifold hypothesis: high-dimensional data lies near a lower-dimensional manifold. Fostering this idea, the authors claim that, if independent factors generate the data, then this manifold is a product manifold: $\mathscr{M} = \mathscr{M}_1 \times \mathscr{M}_2 \times \ldots \times \mathscr{M}_k$, where each $\mathscr{M}_i, i \in \{1 \ldots k\}$ is orthogonal to the others and represents (in the ideal case) at most one generative factor of the data. Given a pair of data (x_1, x_2) that differ in the h-th generative factor only, in [70], their learned representations are considered fully disentangled if they are equal for all projections in the submanifolds $\{\mathscr{M}_i\}_{i=1}^k$, except for the h-th.

In practice, the authors consider a finite-dimensional normed vector space $\mathbf{Z}$ containing the disentangled latent representation, obtained as the output of an encoder network ($\mathbf{Z}$ is indeed a special case of a manifold). Without loss of generality, considering $\mathbf{Z}$ over the field of reals, we can state that $\mathbf{Z} \subseteq \mathbb{R}^d$. Under perfect disentanglement, i.e., a fully minimized loss term $\mathscr{L}$ from Equation 5.2, the latent disentangled representation takes the form of a Cartesian product space $\mathbf{Z} = \mathscr{S}_1 \times \mathscr{S}_2 \times \ldots \times \mathscr{S}_k$, such that for all $(i, j) \in \{1 \ldots k\}, \mathscr{S}_i \cap \mathscr{S}_j = \{0\}$. Intuitively, each subspace encodes an "axis" of variation. Finally, an aggregation step restores the complete latent information, and a decoder g maps the resulting vectors back to the input data space. More specifically, each $\mathbf{S}_i \subseteq \mathscr{R}^d$, such that the subspaces have the same dimensionality as the product space $\mathbf{Z}$. Using a specific regularization (as defined in Equation 5.6), each subspace will have only a few non-zero entries, and the non-zero entries in one subspace will be zero in the others. This results in orthogonal and sparse $\mathbf{S}_i$, which can then be aggregated through summation.

To learn the global manifold structure, a standard autoencoder architecture is used, with f being an encoder that receives non-i.i.d data pairs $(x^{(1)}, x^{(2)})$ and produces the latent representations $(z^{(1)}, z^{(2)})$, and the decoder g that approximates the inverse of f. The pair sampling procedure is designed to induce weak supervision, requiring a pair of images to vary only in one (or a few) generative factors. Additionally, a set of k neural networks $p_i, i \in \{1 \ldots k\}$ called projectors, are trained simultaneously and guided by an unsupervised oracle $\mathscr{O}$, to map the latent codes in the subspaces $\{\mathscr{S}_i\}_{i=1}^k$, each of which contains the corresponding submanifold $\{\mathscr{M}_i\}_{i=1}^k$. To wrap up, the representation framework operates in

the following way:

$$x \xrightarrow{f} z \xrightarrow[i=1\ldots k]{\{p_i\}} \{s_i\} \xrightarrow[i=1\ldots k]{\Sigma_i} \tilde{z} \xrightarrow{g} \tilde{x}\,, \tag{5.1}$$

where the p_is are nonlinear operators, and $\tilde{z}$ and $\tilde{x}$ are the aggregated latent representation and the reconstructed input, respectively. The visual representation of this process is depicted in Figure 5.1, inside the red rectangle. Notably, f and g are initially trained only to minimize the reconstruction error, which is needed to generate the global structure of the manifold $\mathscr{M}$. After a warm-up period, three constraints are added to the optimization problem, each with its own non-learned weight (i.e., β_1, β_2, and β_3, used to regulate the evolution and combination of the losses) to disentangle the latent code:

$$\mathscr{L} = \mathscr{L}_{rec} + \beta_1(\mathscr{L}_{dist} + \mathscr{L}_{spar}) + \beta_2\mathscr{L}_{cons} + \beta_3\mathscr{L}_{reg}\,. \tag{5.2}$$

Specifically, $\mathscr{L}_{rec}$ corresponds to a *reconstruction loss*, implemented in practice as the squared error between the input and the reconstruction following the aggregation operation in the latent space. It is defined as:

$$\mathscr{L}_{rec} = \|x - \bar{x}\|_2^2\,, \tag{5.3}$$

$$\bar{x} = g\left(\sum(p_1(f(x)), \ldots, p_k(f(x))\right)\,. \tag{5.4}$$

This term is necessary to learn the global structure of the manifold $\mathscr{M}$.

The *distance loss*, $\mathscr{L}_{dist}$, is a contrastive loss term that follows an oracle $\mathscr{O}$, which calculates the subspace $\mathscr{S}_i$ where the projections of the images in the pair (x_1, x_2) differ the most, and encourages the projection representation of the two input images onto the subspaces not selected by $\mathscr{O}$ to be as close as possible, while the projections in the selected $\mathscr{S}_i$ to differ the most. It is defined as:

$$\mathscr{L}_{dis} = \sum_{i=1}^{k}(1-\lambda_i)\delta_i^2 + \lambda_i \max(m-\delta_i, 0)^2\,, \tag{5.5}$$

where $\lambda_i = 1$ if $\mathscr{O}(z^{(1)}, z^{(2)}) = i$ and 0 otherwise, while m is a hyperparameter that constrains the points to be at least at a distance m from each other.

$\mathscr{L}_{spar}$ is a *L1* constraint which promotes sparsity and orthogonality between the subspaces. It is defined as:

$$\mathscr{L}_{spar} = \sum_{i=1}^{k} \|p_i(f(x)) \odot \sum_{j\neq i}^{k} p_j(f(x))\|_1\,. \tag{5.6}$$

As mentioned at the beginning of the section, this constraint allows the disentanglement framework to use the sum operation to aggregate the subspaces. The minimization of $\mathscr{L}_{spar}$ promotes sparsity and orthogonality between the subspaces, encouraging each one to have a few non-zero entries that will be zero in the others. In our finite-dimensional setting, this loss is equivalent to imposing that the product space is a direct sum of the subspaces.

$\mathscr{L}_{cons}$, namely the *consistency loss*, encourages each projector p_i to be invariant to changes in subspaces $\mathscr{S}_j, j \neq i$. It is defined as:

$$\mathscr{L}_{cons} = \sum_{i=1}^{k} ||p_i(f_\theta(\hat{x}_{s_i})) - s_i||_2^2 \,, \tag{5.7}$$

with $s_i = p_i f(x_1)$ and $\hat{x}_{s_i} = g(\sum(p_i f(x_1), p_{j\neq i} f(x_2)))$. Along with $\mathscr{L}_{dist}$, this constraint encourages a metric definition of disentanglement, i.e., given a pair of images that are different in image space w.r.t to a particular factor, they should be equally different in the latent representation of that attribute, hosted only in one submanifold which composes the global, product manifold of the latent representation.

Finally, the *regularization loss* $\mathscr{L}_{reg}$ introduces a penalty that ensures the choice of the oracle $\mathscr{O}$ is uniformly distributed among the subspaces, to avoid the collapse of information. This is necessary given the initial warm-up period with only the reconstruction loss being active, as there is no guarantee that information will be equally spread out among the subspaces. It is defined as:

$$\mathscr{L}_{reg} = \sum_{j=1}^{k} \left(\frac{1}{N} \sum_{n=1}^{N} \mathbf{A}_{n,j} - \frac{1}{k} \right)^2 \,, \tag{5.8}$$

with $\mathbf{A} \in \mathbb{R}^{N\times k}$ being the practical implementation of the oracle indicator variables of Equation 5.5 in a batch of N pairs, obtained by applying a weighted softmax to the distance matrix of pairs in each of the k subspaces.

5.4 Methodology

In this Section, after introducing the setting and the adopted notation, we will describe the fundamental contributions of our work: the principal task-based oracle 5.4.1, and the auxiliary task discovery procedure 5.4.2.

Setting and notation We assume the existence of a labeled image dataset $D = \{ (x^{(i)}, y^{(i)}) \,|\, \forall i \in \{1 \dots N\}, x^{(i)} \in \mathbb{R}^{w\times h\times c}, y^{(i)} \in \mathbb{N}\}$, where w is the width, h is the height, and c is the number

of channels, and N is the number of (image, label) tuples. We consider the classification task whose fundamental objective is to learn a mapping from the image space $\{x^{(i)}|\forall i \in \{1\dots N\}\}$ to the corresponding label $\{y^{(i)}|\forall i \in \{1\dots N\}\}$.

5.4.1 The principal task-based oracle

A major issue of the procedure proposed by [70] in our setting is that the oracle will assign the variation given by the principal task label to a random subspace. In order to facilitate the automatic discovery of auxiliary tasks, we must have a way to accommodate the known variation of the principal task in an arbitrary subspace and fix it there, constraining the representation learning. To achieve this, we define a masking procedure that creates the principal task oracle $\hat{\mathscr{O}}$, such that the α-th subspace contains all the variation in the data corresponding to pairs $(x^{(1)},x^{(2)})$ whose elements differ in their label. This implies that we do not inject direct knowledge of the known label, but only whether or not it differs between images of the pair. Thus, in contrast to [70], where all labels are required, we only need to utilize the labels associated with the principal task. To do this, we select a subspace $\alpha \in \{1\dots k\}$ where we wish to force the variation of the principal task labels ($\alpha = 1$ in all our experiments) and define $\hat{\mathscr{O}}$ as:

$$\hat{\mathscr{O}}(z^{(1)},z^{(2)}) = \begin{cases} \alpha & \text{if } y^{(1)} \neq y^{(2)} \\ \arg\max_i d(s_i^{(1)},s_i^{(2)}), i \neq \alpha & \text{otherwise} \end{cases}, \tag{5.9}$$

where $d(s_i^{(1)},s_i^{(2)})$, $i \in \{1...k\}$ is the distance between the projections of the pair $(z^{(1)},z^{(2)})$, in each i-th subspace $\mathscr{S}_i$.

Our new oracle implies that the distance and regularization losses will always force the variation in the data to be encoded in $\mathscr{S}_\alpha$ if $y^{(1)} \neq y^{(2)}$, and in a different subspace otherwise (if $y^{(1)} = y^{(2)}$). The choice of the subspace for the case $y^{(1)} = y^{(2)}$ is made by looking at where the distance between the projections is maximal, as given by the arg max in Equation 5.9. We set $\alpha = 1$ in practice as this is a simple and intuitive choice, but any other value $\in \{1...k\}$ is perfectly suitable. While the organization of the representation space will change with a different choice of the alpha value, this won't affect performance, which will remain the same. In practice, we implement the arg max via a soft max operation on the Euclidean distance normalized by the average length of the vector representation in each subspace. Thanks to the consistency loss, the remaining subspaces can encode other variations while remaining invariant to the ones related to the principal task and contained in $\mathscr{S}_\alpha$. This constraint imposed through the consistency loss will lead us to discover a proper representation where

unknown tasks will correspond to (possibly) multiple subspaces, orthogonal to the ones of the known task.

5.4.2 Auxiliary task discovery

In the disentangled representation of the input data, where the known principal task variation is encoded into a specific subspace, we look to find new, auxiliary tasks in the remaining subspaces.

Intuitively, we wish to have a disentangled subspace that exhibits a clustering tendency over the projected data. Therefore, set p_j to be the projector that has maximized the overall distance d of Equation 5.9 after training and apply a clustering algorithm to the corresponding subspace $\mathscr{S}_j$ to obtain weak labels, determining the new auxiliary classification task. Given that the disentanglement procedure already indicates how much each subspace is divided into clusters via d, choosing subspace $\mathscr{S}_j$ makes it such that the image embeddings are already well separated, giving the clustering procedure that follows an advantage. Under the assumption that tasks living in orthogonal spaces help increase MTL performance [197], we now show why our method regularizes the learning procedure and implicitly guides it towards orthogonal feature spaces for each task.

For the sake of analysis, assume that the model achieves perfect disentanglement. This situation coincides with the condition $\mathscr{L} = 0$ in Equation 5.2. Let $\mathscr{S}_\alpha$ be the subspace that contains the variation of the principal task, forced by Equation 5.9, and $\mathbf{Z} \in \mathbb{R}^{N \times d}$ be the latent representation of the input data x. Then, for any two vectors $\{z^{(1)}, z^{(2)}\} \in \mathbf{Z}$, the functions p_α and p_j will lead to the latent representations $\{s_\alpha^{(1)}, s_j^{(1)}, s_\alpha^{(2)}, s_j^{(2)}\}$, such that the metric d induced by the norm of the space acts independently on $d(s_\alpha^{(1)}, s_\alpha^{(2)})$ from $d(s_j^{(1)}, s_j^{(2)})$, due to the orthogonality of the basis vectors. In simpler terms, the relationship between $d(s_\alpha^{(1)}, s_\alpha^{(2)})$ will not influence the one between $d(s_j^{(1)}, s_j^{(2)})$, given that the information encoded in each subspace is different. This can also be verified in terms of covariance, where due to the orthogonality of subspaces $\langle p_\alpha(\mathbf{Z}), p_j(\mathbf{Z}) \rangle = 0$, which would lead to a covariance (given centered data) of 0. To summarize, the labels of the new auxiliary tasks contain discriminative information that cannot be reduced to the set of the principal task labels. In this way, the auxiliary task will not provide redundant information.

While it is possible to use an arbitrary clustering algorithm, we would like for it to support clusters of arbitrary shapes and for the number of clusters to not be directly specified (e.g., K-Means). Therefore, we utilize HDBSCAN [34] since it allows us to cluster data points based on their proximity and density without explicitly specifying the number of clusters. It is worth noting that HDBSCAN can associate points that cannot be assigned

to any cluster to a "noise" cluster. We retain the data points labeled as belonging to the noise cluster as part of the auxiliary task. If HDBSCAN finds just one cluster, we denote the run as unsuccessful and stop the procedure, given that training a successive MTL model with an auxiliary task with only a single label will lead to a trivial solution. Otherwise, we have discovered a novel task and its corresponding labels $y' \in \mathbb{N}$, which can be used as input to any MTL model, as depicted in the phase 2 of Figure 5.1. In this work, we limit ourselves to finding only one additional task. Scaling on more tasks,i.e., investigating additional subspaces, is the subject of future work. At this stage, we have access to the new dataset $D' = \{ (x_i, y_i, y'_i) \,|\, \forall i \in \{1 \dots N\}, x_i \in \mathbb{R}^{w \times h \times c}, y_i, y'_i \in \mathbb{N}\}$. We are now ready to learn on D' using multi-task classification.

5.5 Experiments

Implementation details We fix the batch size to 32 and the learning rate to 0.0005 for all our experiments. Our code is written within the PyTorch Lightning framework, we use AdamW [161] as an optimizer, and all the experiments are executed on an NVIDIA RTX 3090. We train the disentanglement model for 40 epochs on 3D Shapes and 400 epochs on FACES, CIFAR-10, SVHN, and Cars. The first quarter (25%) of the epochs is used as a warm-up period where only $\mathscr{L}_{rec}$ is active. On the other hand, the coefficients $\beta_1, \beta_2, \beta_3$ follow an exponential warm-up routine after this reconstruction-only phase, such that the constraints they modulate are more gently in the optimization procedure. The projectors $p_i, i \in \{1 \dots k\}$ are implemented as two layers MLPs. Finally, all the MTL models were trained for 150 epochs.

We start by providing a motivating toy example on the 3D Shapes [31] synthetic dataset in 5.5.1. Next, 5.5.2 explains the experiments on real-world image datasets (i.e., FACES [61], CIFAR-10 [130], SVHN [190], and Cars [128]), to cope with real and complex use cases. Finally, in 5.5.3 we discuss some additional experiments and research questions that pinpoint the advantages of our solution.

For all our experiments, we fix the batch size to 32, the learning rate to 0.0005, AdamW [161] as optimizer, within the PyTorch Lightning framework, on an NVIDIA RTX 3090. We train our disentanglement model for 40 epochs on 3D Shapes and 400 epochs on the other datasets. Instead, all the MTL models were trained for 150 epochs.

5.5.1 Synthetic data

To showcase the capabilities of our methodology, we begin our experiments with the 3D Shapes dataset, a widely used benchmark in the disentanglement literature [122, 159, 70]. 3D Shapes is composed of six generative factors: floor hue, wall hue, object hue, scale, shape, and orientation, resulting in 480,000 images. To adapt it to our specific case, we treat the classification of one generative factor as the principal task and pretend to have no knowledge of the others.

Due to the synthetic nature of the images in 3D Shapes, solving classification tasks with a neural network can be excessively easy, leaving limited possibility for improvement through MTL. Specifically, using a simple VGG16 model, we achieve perfect accuracy on each of the six possible tasks, with the expectation of object hue, which is classified with 99.98% accuracy. Thus, to render this setting slightly more complicated, we add salt-and-pepper noise to 15% of the image pixels. With the presence of noise, the classification of the object scale (4 classes) becomes challenging. Hence, we have chosen it as the primary task for our experiments.

We sample pairs of images with 0.5 probability of having the same principal task label and fed these into the disentanglement model, where the encoder f and the decoder g are parametrized through a simple LeNet-like autoencoder architecture. The number of subspaces k is set to 10 as used in [70].

As described Sec. in 5.4.2, we cluster the most disentangled subspace (not considering the forced one) according to the disentanglement loss. The minimum cluster size hyperparameter of HDBSCAN is set to 2% of the number of data points N. In our experiment, it coincides with the subspace which contains the information regarding the object hue (10 classes). Given the optimal disentanglement on 3D Shapes, the auxiliary labels generated by the clustering procedure almost perfectly match the ground-truth object hue labels, having homogeneity and completeness scores of 0.999.

We feed the noisy 3D Shapes and the enriched label set into an MTL hard parameter-sharing architecture with a VGG16 [219] as the backbone and compare Single-Task Learning (STL) vs MTL (using, respectively, one vs two classification heads). For this comparison, we need to perform a train-test split on 3D Shapes, which is non-trivial since the possible combinations of the latent factors in the dataset are present exactly once. Therefore, we split the dataset based on the floor and wall hue labels, allocating the images that contain 5 out of the 10 values for both factors only to the testing set, resulting in a 75-25 train-test split. On the principal task, MTL achieves an accuracy of 0.889, outperforming with a large margin the 0.125 obtained by STL.

Table 5.1 Classification accuracy on the FACES, CIFAR-10, SVHN, and Cars datasets. (*) indicates that the results are the ones reported in the original paper. Boldface indicates the best results, underlined text indicates the models that outperform STL.

Learning Paradigm	FACES	CIFAR-10	SVHN	Cars
STL	0.915	0.844	0.956	0.711
Ours MTL-HPS	0.951	0.848	0.954	0.789
Ours NDDR	0.932	0.872	0.952	0.712
Ours MTI	**0.978**	**0.910**	**0.961**	**0.807**
MAXL	0.933	0.868	0.953	0.638
AuxiLearn	0.915	0.811	0.943	0.644*

5.5.2 Real data

As in the toy example, during the disentanglement procedure, pairs of images are sampled only based on the principal task labels. In FACES this corresponds with the person's facial expression. In CIFAR-10, SVHN, and Cars with the only annotated labels. To obtain a higher fidelity reconstruction, we utilize a ResNet-18 encoder-decoder architecture. The number of subspaces k is set to 10.

During the auxiliary task discovery, we set the minimum cluster size hyperparameter of HDBSCAN to 1% of the number of data points N for all the datasets.

We compare our approach to two different auxiliary learning methods, i.e., MAXL [153] and AuxiLearn [188]. Unlike these auxiliary learning architectures, our discovered auxiliary task can be exploited interchangeably with any MTL model. To have as much control as possible over the experiments, we first use parameter-sharing MTL networks as competitors. Specifically, we select: *i)* HPS (weighing the losses to give more importance to the main task, as explained in [121]), *ii)* NDDR [72], and *iii)* MTI [239], to foster this property of our approach. All these models have a loss term composed of a summation of each task's classification losses. In this way, during the backpropagation, the gradient alters any shared parameters between the two tasks while looking to maximize performance on both, which is what drives the improved generalization capability.

In particular, MTI was proposed to operate with an HRNet backbone [250]. This type of network performs multi-resolution fusion, starting from a high-resolution convolution stream and gradually adding high-to-low-resolution convolution streams one by one, finally connecting the multi-resolution streams in parallel. Since the datasets we operate on contain mostly low-resolution images (e.g., 64×64), learning an HRNet from scratch results in low-quality representations. To avoid this initial problem, we use the official code of MTI, which uses an HRNet pre-trained on ImageNet.

Tab. 5.1 summarizes the results. MTI with our generated labels displays the best performance. Furthermore, even simple CNN-based architecture like HPS and NDDR, achieve superior results compared to MAXL and AuxiLearn. Most notably, we outperform STL with at least one of the MTL models in all the datasets, whereas MAXL and AuxiLearn have significant performance discrepancies between the datasets.

We would like to report, for the sake of completeness, that on the CARS dataset, which contains very complex images, we exploit pre-trained MTL models. For the disentanglement phase, we change the structure of the encoder f such that it does not produce a dense representation in the bottleneck layer but a compressed feature map. The latent space projectors are then learned using 1×1 convolution, and the disentanglement losses are applied to the flattened feature map. Furthermore, in Tab. 5.1, with *, we denote the result we take from the original paper, as we encountered challenges in replicating the performance using the available code and information.

5.5.3 Ablation studies

Is disentanglement useful for discovering new auxiliary tasks?

With this ablation, we wish to quantitatively and qualitatively show how disentanglement is effective in extracting task labels from the underlying data structure. On the FACES dataset, we compare the auxiliary task generated by *Detaux* with the auxiliary task resulting from the clustering on the latent space of an autoencoder that only learns to reconstruct. Without the disentanglement, HPS can only reach 0.9 accuracy, worse than the 0.915 obtained by STL. This reveals that performing auxiliary task mining on the entangled autoencoder space provides an auxiliary task that is not beenficial to the multi-task network compared to our approach. In Fig. 5.2, we compare the projected messy clusters created from the entangled representation *(a)* and the clear grouping obtained in the disentangled representation space *(b)*.

Is it necessary to return to the image space for MTL?

One may ask why we did not work directly in the latent feature space found by the disentanglement procedure. We did some preliminary experiments in this direction, but they yielded inconclusive results and raised implementation issues that are out of the scope of the contribution of this paper. A reason is that most of the MTL frameworks (e.g., MTL-HPS, NDDR, and MTI) require convolution, which is not well defined in the feature space. Another reason is that *Detaux* works at a representation level, regardless of any classification aim induced by a specific classification framework. Its sole purpose is to reveal, together with the principal

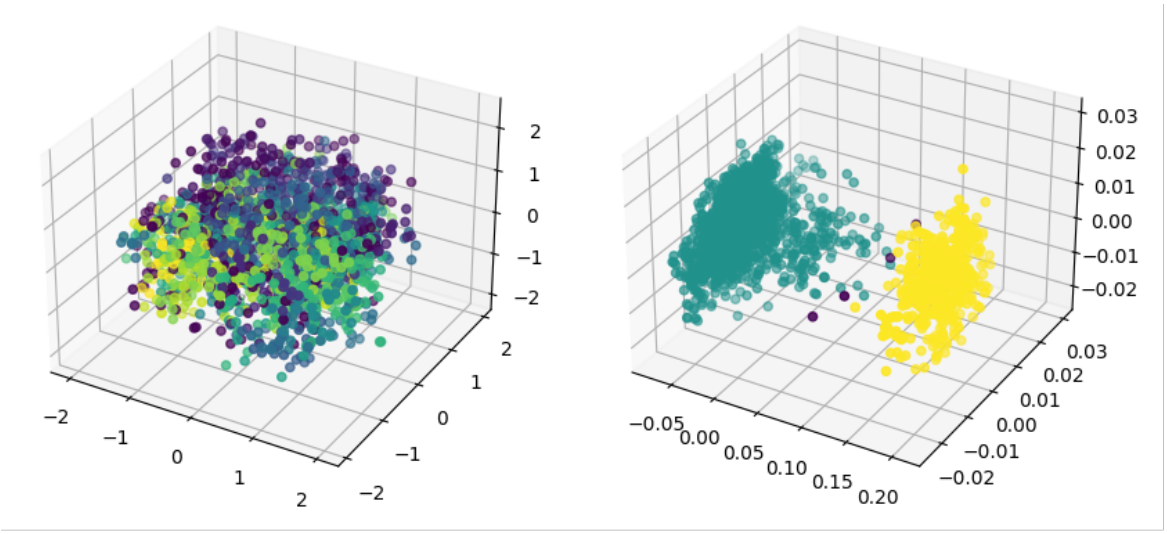


Fig. 5.2 3D visualization (via PCA) of the discovered auxiliary task in the entangled autoencoder feature space *(a)* and the most disentangled subspace *(b)* on FACES. Learning a disentangled representation leads to a subspace that separates the data into two major groups, which correspond to the labels of the new auxiliary task. Instead, using only a reconstruction loss leads to an entangled representation from which it is not beneficial to extract auxiliary tasks. Different colors mean different clusters found by HDBSCAN, which are subsequently projected by PCA in 3 dimensions. Best viewed in color.

subspace determined by the initial labels, other orthogonal complementary subspaces, which can be assumed as tasks if they admit clustering. The output of *Detaux* is an enriched set of labels, that can be exploited with any MTL model. In addition, *Detaux* enables us to visualize and interpret the disentangled subspaces since it reconstructs the images. This procedure allowed us to understand that, in the toy example, the additional task corresponds to the object's hue. Unfortunately, in the more complex real cases, clear interpretation becomes more challenging, barely disclosing in the FACES benchmark the gender as an additional task. We provide an example where this difference is can be visualized via latent interpolation in Fig. 5.3. In the other and much more complex datasets, we had no clue. It is worth noting that our main focus is the framework that transforms a single-task classification problem into an MTL one, with the interpretability of novel tasks being an immediate direction of future work.

Fig. 5.3 An example of latent interpolation in the disentangled subspaces on the FACES dataset. The columns represent a pair of images sampled from the dataset, while the rows represent the chosen number of disentangled subspaces k. The first and last columns hold the real images, the second and second-to-last represent their corresponding reconstructions, and the three middle columns (i.e., columns 3,4,5) represent an interpolation from the left image to the right, with each row being a disentangled subspace. The first row ($\mathscr{S}_\alpha$) contains the principal task, i.e., emotion recognition. One can notice how only the eyes and mouth, related to smiling and being happy are altered, while the rest of the face remains almost identical. In the second row, we can see a candidate auxiliary task, where the gender of the subject seems to change and display different traits. These traits are indeed diverse from the ones dealing with the change in emotion, isolated in the first subspace, showing how we can extract orthogonal auxiliary tasks.

5.6 Conclusion

In this chapter, we revealed a novel outlook on the utility of disentangled representations, utilizing them as a proxy for auxiliary learning in order to improve the accuracy of a principal task originally solvable only in a single-task fashion. Our proposed pipeline facilitates the unsupervised discovery of new tasks from a factorized representation. These newly discovered tasks can be readily incorporated into any MTL framework. We demonstrate empirically that this approach offers advantageous performance, and we analyze various aspects using ablation studies. Our implementation and analysis shed light on the potential of combining disentanglement and MTL for improved performance and generalizability.

Chapter 6

Conclusions

6.1 Overview of the Contributions

This thesis navigates the intricate intersection of deep learning, graphs, and geometry, offering innovative solutions to enhance neural network capabilities in handling complex data structures. Research findings are presented in two different forking paths: First, we show that graph data and deep learning can be leveraged to solve 3D Computer Vision problems, specifically object localization in partially observed 3D scenes and 3D human pose forecasting. The second path delves into more theoretical aspects of Geometric Deep Learning, focusing on the geometry of latent representations learned from neural networks. This intersection of deep learning, graphs, and geometry results in novel solutions that enhance neural networks' capabilities in handling non-Euclidean data structures and learning representations beyond assumed Euclidean latent spaces.

6.2 Limitations

Object localization in partially observed 3D scenes Despite being the first to formalize and propose solutions to this difficult problem, the overall performance of the system is relatively low. Abstract reasoning over space is a very difficult task, and the obtained results reveal that more research is needed before we can adapt object localization in partial scenes in a practical scenario. The proposed approaches model a scene as a graph by considering only a single room. Therefore, a natural step forward would be to consider hierarchical structures such as buildings with multiple rooms. In this scenario, our graph formulation of the problem would not scale well and it would not be able to directly accommodate the hierarchy. Nevertheless, these are issues that can be solved efficiently in future research. The

amount of available data from scene exploration is very large and multimodal. Our proposed approaches work at a very abstract level and do not consider additional multimodal features from the scene such as text embeddings from LLMs or visual embeddings. Multimodal learning is an important future direction for tasks that are related to navigation, such as object localization, but it is something we have not tackled in our current work.

3D Human Pose Forecasting As a model that inherits properties from GCNs, SeS-GCN is prone to over-smoothing, which places an upper bound on the interactions the model can capture, especially in the temporal aspect. We tried to counter this by learning the adjacency matrix of the pose, but we did not consider more expressive alternatives such as modern graph Transformers. Furthermore, after learning the adjacency matrix, our model treats all (graph) neighbors in the same way, a key property of convolution. The soundness of this property for human pose forecasting remains to be explored. As mentioned in the contributions, our model allows us to reach an F1-score of 0.64 on CHICO when it comes to the cobot collision benchmark. Despite reaching state-of-the-art performance, further improvement is needed before our model can be deployed to real-time scenarios. On the other hand, its efficiency allows for immediate use in a hardware-restricted environment, as far as pose forecasting is concerned. Related to the dataset, CHICO currently contains interaction only with type or robotic arm. Even though it is the first dataset of its kind, this puts a limit on the type of trajectories the model can be tested on. While this is not a limitation per se but more so a derivative of modern learning systems, it limits the transferability of methods trained on the proposed dataset in other environments that might not have exactly the same KUKA robotic arm.

Graph-level Representation Learning At the current stage, Graph-JEPA has been designed and tested with regard to graph-level tasks. Therefore, the inability to work on node-level and edge-level tasks is a limitation of the proposed approach, given that there exist models present in the state of the art that can tackle all of these different types of tasks The expressiveness of Graph-JEPA in terms of the WL test is only validated empirically, which does not allow us to directly compare it with (MP)GNNs. As a final limitation and direction of future work, insights into the optimal geometry of the embedding space for graph SSL are not conclusive. We observe that hyperbolic representations are more optimal for graph-level tasks, as conjecture, but the hierarchical structure might not be as necessary for node-level tasks. In this regard, an interesting theoretical direction is to try and characterize if self-supervised and unsupervised deep learning approaches already tend to learn embeddings that are hierarchical and look at the geometry of the latent space. Mixed geometry approaches

might be an important advancement in this regard. Additional empirical validation regarding the scaling performance of Graph-JEPA is necessary, especially on the large-scale OGB datasets.

Auxiliary Multi-task Learning through Geometric Disentanglement The main limitation of Detaux is the fact that the generated auxiliary tasks are classification tasks, given the discretization due to the clustering procedure. As such, our framework cannot currently produce different types of tasks. Another limiting factor is that in order to generate meaningful submanifolds, the initial representation of the product manifold should be as close as possible to a bijection with the input data. In simpler words, a good reconstruction is necessary in the first part of the pipeline, which requires sophisticated models for more complex images. While we have focused on image data, a great open question remains whether it is possible to use the same pipeline to extract auxiliary tasks for dynamic data such as videos. This point is not explored and is left for future work.

6.3 Future Work

There is a massive body of work that is currently exploring the extensions of neural networks in alternative geometries, in particular hyperbolic [71, 208, 174, 218, 87] and (learnable) mixed geometries [279]. One of the most important aspects regarding future work in this domain is computational efficiency, given that arc lengths and distances in non-Euclidean geometries are usually much more expensive. This is partly due to the fact that calculations on the Euclidean plane have been optimized for a very long time in the computer science community. A very important research direction is the geometric understanding of learning capabilities and characterization of the representations learned by neural networks [198, 2]. In this regard, understanding what invariants might allow DL approaches to learn some data while failing to learn others is a very important direction. Additionally, using geometry and even more abstract concepts from topology to enforce valuable properties such as disentanglement [22, 70, 179] is an important research direction.

Regarding graph neural networks, it is vital for the research community to understand how and why self-supervised learning on graphs seems to not work and scale, as well as in other domains. This might be due to the GNN architecture per se, or due to the fact that the self-supervised signals are not simple to adapt to graphs [143]. A promising direction for improving graph learning is extending the concept of physical diffusion (e.g., random walks) as a learning mechanism [18]. In this regard, geometric concepts can play a significant role. Finally, the well-celebrated problems of oversmoothing[25] and oversquashing [234] most

surely present the chances for novel work and breaking into a new era of neural networks on graphs.

6.4 Closing Remarks

I want to conclude this thesis by touching upon something that I consider really important, which is the importance of this research topic in the study of learning systems that can lead to AI. It is a common belief that humans have a geometric understanding of the world. This can be clearly understood when one reflects on the strong connections of geometry with physics, from classical mechanics to quantum mechanics and the theory of relativity [92] [169]. The more critical aspect of this connection is that, often, humans (intuitively and subconsciously) reason about physical concepts through shapes, distances, and sizes. This subtle process encodes a defining feature of intelligent beings, i.e., that they are able to understand and alter their environment to their advantage intuitively. A perfect example is the wheel, considered the most influential invention ever. At the time of its invention in ancient Mesopotamia, people had no conception whatsoever of mechanics, but somehow, they understood from what they had observed during their lifetime that a circular shape would reduce friction and facilitate motion under the application of external forces.

Geometry provides a way to give form to these physical concepts, and the concept of symmetry is what drives most modern physical theories. What is most astounding about this fact is that, albeit at a basic level, human beings are able to understand geometric concepts in a somehow natural manner. If we think about data the same way, geometric understanding becomes an essential factor when it comes to generalization. It is fundamental because if symmetries and invariances in the data can be induced, this implies the learner understands that the structure present in the data, and not just the signals, are needed to satisfy its objective. Based on this conjecture, it should be more straightforward to transfer knowledge to identical structures and understand the similarities between different structures. Thus generalizing over the same "space" becomes simpler. This aspect of Geometric Deep Learning is often not discussed, but it should be held in high regard, and I believe it will contribute to several research directions in the future that deal with the physics of AI systems.

References

[1] Abboud, R., Ceylan, I. I., Grohe, M., and Lukasiewicz, T. (2020). The surprising power of graph neural networks with random node initialization. *arXiv preprint arXiv:2010.01179*.

[2] Acosta, F., Sanborn, S., Duc, K. D., Madhav, M., and Miolane, N. (2023). Quantifying extrinsic curvature in neural manifolds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 610–619.

[3] Adiga, S., Attia, M. A., Chang, W.-T., and Tandon, R. (2018). On the tradeoff between mode collapse and sample quality in generative adversarial networks. In *2018 IEEE global conference on signal and information processing (GlobalSIP)*, pages 1184–1188. IEEE.

[4] Aksan, E., Kaufmann, M., and Hilliges, O. (2019). Structured prediction helps 3d human motion modelling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

[5] Anderson, J. W. (2006). *Hyperbolic Geometry*. Springer London.

[6] Armeni, I., He, Z.-Y., Gwak, J., Zamir, A. R., Fischer, M., Malik, J., and Savarese, S. (2019). 3d scene graph: A structure for unified semantics, 3d space, and camera.

[7] Assran, M., Duval, Q., Misra, I., Bojanowski, P., Vincent, P., Rabbat, M., LeCun, Y., and Ballas, N. (2023). Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15619–15629.

[8] Bachmann, G., Bécigneul, G., and Ganea, O. (2020). Constant curvature graph convolutional networks. In *International conference on machine learning*, pages 486–496. PMLR.

[9] Baevski, A., Hsu, W.-N., Xu, Q., Babu, A., Gu, J., and Auli, M. (2022). Data2vec: A general framework for self-supervised learning in speech, vision and language. In *International Conference on Machine Learning*, pages 1298–1312. PMLR.

[10] Bai, S., Kolter, J. Z., and Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *ArXiv*, abs/1803.01271.

[11] Balcilar, M., Renton, G., Héroux, P., Gaüzère, B., Adam, S., and Honeine, P. (2020). Spectral-designed depthwise separable graph neural networks. In *Proceedings of Thirty-seventh International Conference on Machine Learning (ICML 2020)-Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*.

[12] Bao, J., Duan, N., Zhou, M., and Zhao, T. (2014). Knowledge-based question answering as machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

[13] Bardes, A., Garrido, Q., Ponce, J., Chen, X., Rabbat, M., LeCun, Y., Assran, M., and Ballas, N. (2023a). V-jepa: Latent video prediction for visual representation learning.

[14] Bardes, A., Ponce, J., and LeCun, Y. (2023b). Mc-jepa: A joint-embedding predictive architecture for self-supervised learning of motion and content features. *arXiv preprint arXiv:2307.12698*.

[15] Batra, D., Gokaslan, A., Kembhavi, A., Maksymets, O., Mottaghi, R., Savva, M., Toshev, A., and Wijmans, E. (2020). Objectnav revisited: On evaluation of embodied agents navigating to objects. *arXiv preprint arXiv:2006.13171*.

[16] Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. (2018). Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

[17] Bauer, A., Wollherr, D., and Buss, M. (2008). Human–robot collaboration: a survey. *International Journal of Humanoid Robotics*, 5(01):47–66.

[18] Behmanesh, M., Krahn, M., and Ovsjanikov, M. (2023). Tide: Time derivative diffusion for deep learning on graphs. In *International Conference on Machine Learning*, pages 2015–2030. PMLR.

[19] Beltran, E. P., Diwa, A. A. S., Gales, B. T. B., Perez, C. E., Saguisag, C. A. A., and Serrano, K. K. D. (2018). Fuzzy logic-based risk estimation for safe collaborative robots. In *2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology,Communication and Control, Environment and Management (HNICEM)*, pages 1–5.

[20] Benesova, K., Svec, A., and Suppa, M. (2021). Cost-effective deployment of bert models in serverless environment.

[21] Bengio, Y. (2012). Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 17–36. JMLR Workshop and Conference Proceedings.

[22] Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

[23] Bertasius, G., Wang, H., and Torresani, L. (2021). Is space-time attention all you need for video understanding? In *Proceedings of the International Conference on Machine Learning (ICML)*.

[24] Bertsimas, D. and Tsitsiklis, J. N. (1997). *Introduction to linear optimization*, volume 6. Athena scientific Belmont, MA.

[25] Bo, D., Wang, X., Shi, C., and Shen, H. (2021). Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3950–3957.

[26] Bresson, X. and Laurent, T. (2017). Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*.

[27] Brody, S., Alon, U., and Yahav, E. (2021). How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*.

[28] Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. (1993). Signature verification using a" siamese" time delay neural network. *Advances in neural information processing systems*, 6.

[29] Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. (2021). Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*.

[30] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

[31] Burgess, C. and Kim, H. (2018). 3d shapes dataset.

[32] Bütepage, J., Kjellström, H., and Kragic, D. (2017). Anticipating many futures: Online human motion prediction and synthesis for human-robot collaboration. *ArXiv*, abs/1702.08212.

[33] Cai, Y., Huang, L., Wang, Y., Cham, T.-J., Cai, J., Yuan, J., Liu, J., Yang, X., Zhu, Y., Shen, X., Liu, D., Liu, J., and Thalmann, N. M. (2020). Learning progressive joint propagation for human motion prediction. In *The European Conference on Computer Vision (ECCV)*.

[34] Campello, R. J., Moulavi, D., and Sander, J. (2013). Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer.

[35] Cao, Z., Hidalgo Martinez, G., Simon, T., Wei, S., and Sheikh, Y. A. (2019). Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[36] Caruana, R. (1997). Multitask learning. *Machine learning*, 28:41–75.

[37] Castro, A., Silva, F., and Santos, V. (2021). Trends of human-robot collaboration in industry contexts: Handover, learning, and metrics. *Sensors*, 21(12):4113.

[38] Chaplot, D. S., Gandhi, D., Gupta, A., and Salakhutdinov, R. (2020). Object goal navigation using goal-oriented semantic exploration. In *Proceedings of Neural Information Processing Systems (NeurIPS)*.

[39] Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., and Sun, X. (2020a). Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

[40] Chen, J. H. and Song, K. T. (2018). Collision-Free Motion Planning for Human-Robot Collaborative Safety under Cartesian Constraint. *IEEE Int. Conf. Robot. Autom.*, pages 4348–4354.

[41] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020b). A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

[42] Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016). Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *Advances in neural information processing systems*, 29.

[43] Chen, X. and He, K. (2021). Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15750–15758.

[44] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1800–1807.

[45] Colom, R., Karama, S., Jung, R. E., and Haier, R. J. (2010). Human intelligence and brain networks. *Dialogues in clinical neuroscience*, 12(4):489–501.

[46] Costanzo, M., De Maria, G., Lettera, G., and Natale, C. (2021). A multimodal approach to human safety in collaborative robotic workcells. *IEEE Transactions on Automation Science and Engineering*, PP:1–15.

[47] Cui, Q., Sun, H., and Yang, F. (2020). Learning dynamic relationships for 3d human motion prediction. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6518–6526.

[48] Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T., and Nießner, M. (2017). Scannet: Richly-annotated 3d reconstructions of indoor scenes.

[49] Dallel, M., Havard, V., Baudry, D., and Savatier, X. (2020). Inhard - industrial human action recognition dataset in the context of industrial collaborative robotics. In *2020 IEEE International Conference on Human-Machine Systems (ICHMS)*.

[50] Dang, L., Nie, Y., Long, C., Zhang, Q., and Li, G. (2021). MSR-GCN: Multi-scale residual graph convolution networks for human motion prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

[51] Dawid, A. and LeCun, Y. (2023). Introduction to latent variable energy-based models: A path towards autonomous machine intelligence. *arXiv preprint arXiv:2306.02572*.

[52] Deco, G., Vidaurre, D., and Kringelbach, M. (2021). Revisiting the global workspace orchestrating the hierarchical organization of the human brain. *Nature Human Behaviour*, 5:497 – 511.

[53] Dery, L. M., Michel, P., Khodak, M., Neubig, G., and Talwalkar, A. (2022). Aang: Automating auxiliary learning. In *The Eleventh International Conference on Learning Representations*.

[54] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

[55] Dhamo, H., Manhardt, F., Navab, N., and Tombari, F. (2021). Graph-to-3d: End-to-end generation and manipulation of 3d scenes using scene graphs.

[56] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

[57] Duarte, N. F., Raković, M., Tasevski, J., Coco, M. I., Billard, A., and Santos-Victor, J. (2018). Action anticipation: Reading the intentions of humans and robots. *IEEE Robotics and Automation Letters*, 3(4):4132–4139.

[58] Dwivedi, V. P. and Bresson, X. (2020). A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*.

[59] Dwivedi, V. P., Luu, A. T., Laurent, T., Bengio, Y., and Bresson, X. (2021). Graph neural networks with learnable structural and positional representations. *arXiv preprint arXiv:2110.07875*.

[60] Eastwood, C. and Williams, C. K. (2018). A framework for the quantitative evaluation of disentangled representations. In *International Conference on Learning Representations*.

[61] Ebner, N. C., Riediger, M., and Lindenberger, U. (2010). Faces—a database of facial expressions in young, middle-aged, and older women and men: Development and validation. *Behavior research methods*, 42:351–362.

[62] Elmannai, W. and Elleithy, K. (2017). Sensor-based assistive devices for visually-impaired people: current status, challenges, and future directions. *Sensors*, 17(3):565.

[63] Faldu, K., Sheth, A., Kikani, P., and Akabari, H. (2021). Ki-bert: Infusing knowledge context for better language and domain understanding. *arXiv preprint arXiv:2104.08145*.

[64] Fei, Z., Fan, M., and Huang, J. (2023). A-jepa: Joint-embedding predictive architecture can listen. *arXiv preprint arXiv:2311.15830*.

[65] Fey, M. and Lenssen, J. E. (2019). Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

[66] Fieraru, M., Zanfir, M., Oneata, E., Popa, A.-I., Olaru, V., and Sminchisescu, C. (2020). Three-dimensional reconstruction of human interactions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7214–7223.

[67] Fifty, C., Amid, E., Zhao, Z., Yu, T., Anil, R., and Finn, C. (2021). Efficiently identifying task groupings for multi-task learning. *Advances in Neural Information Processing Systems*, 34:27503–27516.

[68] Fragkiadaki, K., Levine, S., Felsen, P., and Malik, J. (2015). Recurrent network models for human dynamics. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4346–4354.

[69] Fumero, M., Cosmo, L., Melzi, S., and Rodola, E. (2021a). Learning disentangled representations via product manifold projection. In *Proceedings of the 38th International Conference on Machine Learning*, pages 3530–3540. PMLR. ISSN: 2640-3498.

[70] Fumero, M., Cosmo, L., Melzi, S., and Rodolà, E. (2021b). Learning disentangled representations via product manifold projection. In *International conference on machine learning*, pages 3530–3540. PMLR.

[71] Ganea, O.-E., Bécigneul, G., and Hofmann, T. (2018). Hyperbolic Neural Networks. arXiv:1805.09112 [cs, stat].

[72] Gao, Y., Ma, J., Zhao, M., Liu, W., and Yuille, A. L. (2019). Nddr-cnn: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3205–3214.

[73] Garcia-Esteban, J. A., Piardi, L., Leitao, P., Curto, B., and Moreno, V. (2021). An interaction strategy for safe human Co-working with industrial collaborative robots. *Proc. - 2021 4th IEEE Int. Conf. Ind. Cyber-Physical Syst. ICPS 2021*, pages 585–590.

[74] Gay, P., Stuart, J., and Del Bue, A. (2018). Visual graphs from motion (vgfm): Scene understanding with object geometry reasoning. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*.

[75] Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. In *The International Conference on Machine Learning (ICML)*.

[76] Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR.

[77] Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

[78] Giuliari, F., Castellini, A., Berra, R., Bue, A. D., Farinelli, A., Cristani, M., Setti, F., and Wang, Y. (2021). Pomp++: Pomcp-based active visual search in unknown indoor environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[79] Giuliari, F., Skenderi, G., Cristani, M., Del Bue, A., and Wang, Y. (2023). Leveraging commonsense for object localisation in partial scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[80] Giuliari, F., Skenderi, G., Cristani, M., Wang, Y., and Del Bue, A. (2022). Spatial commonsense graph for object localisation in partial scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19518–19527.

[81] Gopalakrishnan, A., Mali, A., Kifer, D., Giles, L., and Ororbia, A. G. (2019). A neural temporal model for human motion prediction. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12108–12117.

[82] Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. (2020). Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284.

[83] Gu, J., Joty, S., Cai, J., Zhao, H., Yang, X., and Wang, G. (2019a). Unpaired image captioning via scene graph alignments.

[84] Gu, J., Zhao, H., Lin, Z. L., Li, S., Cai, J., and Ling, M. (2019b). Scene graph generation with external knowledge and image reconstruction.

[85] Gualtieri, L., Palomba, I., Wehrle, E. J., and Vidoni, R. (2020). *The Opportunities and Challenges of SME Manufacturing Automation: Safety and Ergonomics in Human–Robot Collaboration*. Springer International Publishing.

[86] Guo, W., Bie, X., Alameda-Pineda, X., and Moreno-Noguer, F. (2021). Multi-person extreme motion prediction with cross-interaction attention. *arXiv preprint arXiv:2105.08825*.

[87] Guo, Y., Guo, H., and Yu, S. X. (2022). Co-sne: Dimensionality reduction and visualization for hyperbolic data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21–30.

[88] Gupta, K., Lazarow, J., Achille, A., Davis, L. S., Mahadevan, V., and Shrivastava, A. (2021). Layouttransformer: Layout generation and completion with self-attention.

[89] Gutmann, M. and Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 297–304. JMLR Workshop and Conference Proceedings.

[90] Gutteridge, B., Dong, X., Bronstein, M. M., and Di Giovanni, F. (2023). Drew: Dynamically rewired message passing with delay. In *International Conference on Machine Learning*, pages 12252–12267. PMLR.

[91] Haddadin, S., Albu-Schaffer, A., Frommberger, M., Rossmann, J., and Hirzinger, G. (2009). The "dlr crash report": Towards a standard crash-testing protocol for robot safety-part i: Results. In *2009 IEEE International Conference on Robotics and Automation*, pages 272–279. IEEE.

[92] Halliday, D., Resnick, R., and Walker, J. (2013). *Fundamentals of physics*. John Wiley & Sons.

[93] Hanin, B. (2019). Universal function approximation by deep neural nets with bounded width and relu activations. *Mathematics*, 7(10):992.

[94] Hassani, K. and Khasahmadi, A. H. (2020). Contrastive multi-view representation learning on graphs. In *International conference on machine learning*, pages 4116–4126. PMLR.

[95] He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. (2022). Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009.

[96] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

[97] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

[98] He, X., Hooi, B., Laurent, T., Perold, A., LeCun, Y., and Bresson, X. (2023). A generalization of vit/mlp-mixer to graphs. In *International Conference on Machine Learning*, pages 12724–12745. PMLR.

[99] Higgins, I., Amos, D., Pfau, D., Racaniere, S., Matthey, L., Rezende, D., and Lerchner, A. (2018). Towards a definition of disentangled representations. *arXiv preprint arXiv:1812.02230*.

[100] Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2017). beta-vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*.

[101] Hinton, G., Dean, J., and Vinyals, O. (2014). Distilling the knowledge in a neural network. In *NIPS*, pages 1–9.

[102] Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800.

[103] Hinton, G. E. and Zemel, R. (1993). Autoencoders, minimum description length and helmholtz free energy. *Advances in neural information processing systems*, 6.

[104] Hitchman, M. P. (2009). *Geometry with an introduction to cosmic topology*. Jones & Bartlett Learning.

[105] Hjorth, S. and Chrysostomou, D. (2022). Human–robot collaboration in industrial environments: A literature review on non-destructive disassembly. *Robotics and Computer-Integrated Manufacturing*, 73:102–208.

[106] Horan, D., Richardson, E., and Weiss, Y. (2021). When is unsupervised disentanglement possible? *Advances in Neural Information Processing Systems*, 34:5150–5161.

[107] Hou, Z., Liu, X., Cen, Y., Dong, Y., Yang, H., Wang, C., and Tang, J. (2022). Graphmae: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 594–604.

[108] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications.

[109] Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., and Leskovec, J. (2019). Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*.

[110] Hu, Z., Dong, Y., Wang, K., Chang, K.-W., and Sun, Y. (2020). Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1857–1867.

[111] Ionescu, C., Papava, D., Olaru, V., and Sminchisescu, C. (2014). Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7).

[112] ISO (2021). ISO/TS 15066:2016. Robots and robotic devices — Collaborative robots. https://www.iso.org/obp/ui/#iso:std:iso:ts:15066:ed-1:v1:en.

[113] Jain, A., Zamir, A. R., Savarese, S., and Saxena, A. (2016). Structural-rnn: Deep learning on spatio-temporal graphs. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5308–5317.

[114] Jayaraman, D., Sha, F., and Grauman, K. (2014). Decorrelating semantic visual attributes by resisting the urge to share. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1629–1636.

[115] Jin, W., Derr, T., Liu, H., Wang, Y., Wang, S., Liu, Z., and Tang, J. (2020). Self-supervised learning on graphs: Deep insights and new direction. *arXiv preprint arXiv:2006.10141*.

[116] Jin, W., Liu, X., Zhao, X., Ma, Y., Shah, N., and Tang, J. (2021). Automated self-supervised learning for graphs. In *International Conference on Learning Representations*.

[117] Johnson, J., Krishna, R., Stark, M., Li, L.-J., Shamma, D. A., Bernstein, M. S., and Fei-Fei, L. (2015). Image retrieval using scene graphs.

[118] Kanazawa, A., Kinugawa, J., and Kosuge, K. (2019). Adaptive Motion Planning for a Collaborative Robot Based on Prediction Uncertainty to Enhance Human Safety and Work Efficiency. *IEEE Trans. Robot.*, 35(4):817–832.

[119] Kang, S., Kim, M., and Kim, K. (2019). Safety Monitoring for Human Robot Collaborative Workspaces. *Int. Conf. Control. Autom. Syst.*, 2019-October(Iccas):1192–1194.

[120] Karypis, G. and Kumar, V. (1998). A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392.

[121] Kendall, A., Gal, Y., and Cipolla, R. (2018). Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491.

[122] Kim, H. and Mnih, A. (2018). Disentangling by factorising. In *International Conference on Machine Learning*, pages 2649–2658. PMLR.

[123] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[124] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[125] Kipf, T. N. and Welling, M. (2016a). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

[126] Kipf, T. N. and Welling, M. (2016b). Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.

[127] Knudsen, M. and Kaivo-oja, J. (2020). Collaborative robots: Frontiers of current literature. *Journal of Intelligent Systems: Theory and Applications*, 3:13–20.

[128] Krause, J., Stark, M., Deng, J., and Fei-Fei, L. (2013). 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561.

[129] Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.-J., Shamma, D. A., Bernstein, M. S., and Fei-Fei, L. (2016). Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123:32–73.

[130] Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

[131] Lai, G., Liu, H., and Yang, Y. (2018). Learning graph convolution filters from data manifold.

[132] Laplaza, J., Pumarola, A., Moreno-Noguer, F., and Sanfeliu, A. (2021). Attention deep learning based model for predicting the 3d human body pose using the robot human handover phases. In *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*, pages 161–166. IEEE.

[133] LeCun, V., Denker, J., and Solla, S. (1989). Optimal brain damage. In *Advances in Neural Information Processing Systems*.

[134] LeCun, Y. (2022). A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62.

[135] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

[136] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[137] Lee, M. (2023). The geometry of feature space in deep learning models: A holistic perspective and comprehensive review. *Mathematics*, 11(10):2375.

[138] Lee, N., Lee, J., and Park, C. (2022). Augmentation-free self-supervised learning on graphs. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 7372–7380.

[139] Lee, S., Kim, J.-W., Oh, Y., and Jeon, J. H. (2019). Visual question answering over scene graph. In *Proceedings of the First International Conference on Graph Computing (GC)*.

[140] Lemmerz, K., Glogowski, P., Kleineberg, P., Hypki, A., and Kuhlenkötter, B. (2019). A Hybrid Collaborative Operation for Human-Robot Interaction Supported by Machine Learning. *Int. Conf. Hum. Syst. Interact. HSI*, 2019-June:69–75.

[141] Li, C., Zhang, Z., Sun Lee, W., and Hee Lee, G. (2018). Convolutional sequence to sequence model for human dynamics. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[142] Li, G., Su, H., and Zhu, W. (2017). Incorporating external knowledge to answer open-domain visual questions with dynamic memory networks. *arXiv preprint arXiv:1712.00733*.

[143] Li, J., Wu, R., Sun, W., Chen, L., Tian, S., Zhu, L., Meng, C., Zheng, Z., and Wang, W. (2023). What's behind the mask: Understanding masked graph modeling for graph autoencoders. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1268–1279.

[144] Li, M., Chen, S., Zhao, Y., Zhang, Y., Wang, Y., and Tian, Q. (2020a). Dynamic multiscale graph neural networks for 3d skeleton based human motion prediction. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 211–220.

[145] Li, M., Lin, J., Ding, Y., Liu, Z., Zhu, J.-Y., and Han, S. (2020b). Gan compression: Efficient architectures for interactive conditional gans. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5283–5293.

[146] Li, M., Patil, A. G., Xu, K., Chaudhuri, S., Khan, O., Shamir, A., Tu, C., Chen, B., Cohen-Or, D., and Zhang, H. (2019). Grains: Generative recursive autoencoders for indoor scenes. *ACM Transactions on Graphics (TOG)*, 38(2):1–16.

[147] Li, X. and Li, D. (2021). Gpfs: A graph-based human pose forecasting system for smart home with online learning. *ACM Trans. Sen. Netw.*, 17(3).

[148] Li, Y. and Shan, S. (2021). Meta auxiliary learning for facial action unit detection. *IEEE Transactions on Affective Computing*.

[149] Liebel, L. and Körner, M. (2018). Auxiliary tasks in multi-task learning. *arXiv preprint arXiv:1805.06334*.

[150] Lim, J., Lee, J., Lee, C., Kim, G., Cha, Y., Sim, J., and Rhim, S. (2021). Designing path of collision avoidance for mobile manipulator in worker safety monitoring system using reinforcement learning. *ISR 2021 - 2021 IEEE Int. Conf. Intell. Saf. Robot.*, pages 94–97.

[151] Liu, C., Zheng, C.-T., Qian, S., Wu, S., and Wong, H.-S. (2019a). Encoding sparse and competitive structures among tasks in multi-task learning. *Pattern Recognition*, 88:689–701.

[152] Liu, J., Yang, M., Zhou, M., Feng, S., and Fournier-Viger, P. (2022). Enhancing hyperbolic graph embeddings via contrastive learning. *arXiv preprint arXiv:2201.08554*.

[153] Liu, S., Davison, A., and Johns, E. (2019b). Self-supervised generalisation with meta auxiliary learning. *Advances in Neural Information Processing Systems*, 32.

[154] Liu, S. and Deng, W. (2015). Very deep convolutional neural network based image classification using small training sample size. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 730–734.

[155] Liu, X., Zhang, F., Hou, Z., Mian, L., Wang, Z., Zhang, J., and Tang, J. (2023). Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):857–876.

[156] Loaiza-Ganem, G., Ross, B. L., Cresswell, J. C., and Caterini, A. L. (2022). Diagnosing and fixing manifold overfitting in deep generative models. *Transactions on Machine Learning Research*.

[157] Locatello, F., Bauer, S., Lucic, M., Raetsch, G., Gelly, S., Schölkopf, B., and Bachem, O. (2019a). Challenging common assumptions in the unsupervised learning of disentangled representations. In *international conference on machine learning*, pages 4114–4124. PMLR.

[158] Locatello, F., Poole, B., Rätsch, G., Schölkopf, B., Bachem, O., and Tschannen, M. (2020a). Weakly-supervised disentanglement without compromises. In *International Conference on Machine Learning*, pages 6348–6359. PMLR.

[159] Locatello, F., Tschannen, M., Bauer, S., Rätsch, G., Schölkopf, B., and Bachem, O. (2019b). Disentangling factors of variation using few labels. *arXiv preprint arXiv:1905.01258*.

[160] Locatello, F., Tschannen, M., Bauer, S., Rätsch, G., Schölkopf, B., and Bachem, O. (2020b). Disentangling factors of variation using few labels. *arXiv 1905.01258*.

[161] Loshchilov, I. and Hutter, F. (2018). Decoupled weight decay regularization. In *International Conference on Learning Representations*.

[162] Luo, A., Zhang, Z., Wu, J., and Tenenbaum, J. B. (2020). End-to-end optimization of scene layout.

[163] Lyons, D. W. (2023). *Introduction to Groups and Geometries*. LibreTexts Mathematics.

[164] López, F., Pozzetti, B., Trettel, S., Strube, M., and Wienhard, A. (2021). Symmetric Spaces for Graph Embeddings: A Finsler-Riemannian Approach. arXiv:2106.04941 [cs].

[165] Ma, Y. and Tang, J. (2021a). *Deep learning on graphs*. Cambridge University Press.

[166] Ma, Y. and Tang, J. (2021b). *Deep Learning on Graphs*. Cambridge University Press.

[167] Magrini, E., Ferraguti, F., Ronga, A. J., Pini, F., De Luca, A., and Leali, F. (2020). Human-robot coexistence and interaction in open industrial cells. *Robotics and Computer-Integrated Manufacturing*, 61.

[168] Mahmood, N., Ghorbani, N., Troje, N. F., Pons-Moll, G., and Black, M. J. (2019). AMASS: Archive of motion capture as surface shapes. In *International Conference on Computer Vision*.

[169] Maia, M. D. (2011). *Geometry of the Fundamental Interactions: On Riemann's Legacy to High Energy Physics and Cosmology*. Springer Science & Business Media.

[170] Mao, W., Liu, M., and Salzmann, M. (2020). History repeats itself: Human motion prediction via motion attention. In *The European Conference on Computer Vision (ECCV)*.

[171] Mao, W., Liu, M., Salzmann, M., and Li, H. (2019). Learning trajectory dependencies for human motion prediction. In *The IEEE International Conference on Computer Vision (ICCV)*.

[172] Martinez, J., Black, M. J., and Romero, J. (2017). On human motion prediction using recurrent neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[173] Mathieu, E., Lan, C. L., Maddison, C. J., Tomioka, R., and Teh, Y. W. (2019a). Continuous Hierarchical Representations with Poincar\'e Variational Auto-Encoders. arXiv:1901.06033 [cs, stat].

[174] Mathieu, E., Le Lan, C., Maddison, C. J., Tomioka, R., and Teh, Y. W. (2019b). Continuous hierarchical representations with poincaré variational auto-encoders. *Advances in neural information processing systems*, 32.

[175] Matthias, B. and Reisinger, T. (2016). Example application of ISO/TS 15066 to a collaborative assembly scenario. *47th Int. Symp. Robot. ISR 2016*, 2016:88–92.

[176] Maziarka, Ł., Nowak, A., Wołczyk, M., and Bedychaj, A. (2023). On the relationship between disentanglement and multi-task learning. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2022, Grenoble, France, September 19–23, 2022, Proceedings, Part I*, pages 625–641. Springer.

[177] Meng, Q., Catchpoole, D., Skillicom, D., and Kennedy, P. J. (2017). Relational autoencoder for feature extraction. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 364–371.

[178] Meng, Q., Pawlowski, N., Rueckert, D., and Kainz, B. (2019). Representation disentanglement for multi-task learning with application to fetal ultrasound. In *Smart Ultrasound Imaging and Perinatal, Preterm and Paediatric Image Analysis*, pages 47–55. Springer.

[179] Mercatali, G., Freitas, A., and Garg, V. (2022). Symmetry-induced Disentanglement on Graphs.

[180] Michalos, G., Makris, S., Tsarouchi, P., Guasch, T., Kontovrakis, D., and Chryssolouris, G. (2015). Design considerations for safe human-robot collaborative workplaces. *Procedia CIrP*, 37:248–253.

[181] Minelli, M., Sozzi, A., De Rossi, G., Ferraguti, F., Setti, F., Muradore, R., Bonfè, M., and Secchi, C. (2020). Integrating model predictive control and dynamic waypoints generation for motion planning in surgical scenario. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3157–3163.

[182] Molchanov, P., Tyree, S., Karras, T., Aila, T., and Kautz, J. (2017). Pruning convolutional neural networks for resource efficient inference.

[183] Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., and Neumann, M. (2020). Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*.

[184] Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. (2019). Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4602–4609.

[185] Moutsinas, G., Shuaib, C., Guo, W., and Jarvis, S. (2021). Graph hierarchy: a novel framework to analyse hierarchical structures in complex networks. *Scientific Reports*, 11(1):13943.

[186] Nam, J., Tack, J., Lee, K., Lee, H., and Shin, J. (2023). STUNT: Few-shot tabular learning with self-generated tasks from unlabeled tables. In *The Eleventh International Conference on Learning Representations*.

[187] Nascimento, H., Mujica, M., and Benoussaad, M. (2020). Collision avoidance in human-robot interaction using kinect vision system combined with robot's model and data. *IEEE Int. Conf. Intell. Robot. Syst.*, pages 10293–10298.

[188] Navon, A., Achituve, I., Maron, H., Chechik, G., and Fetaya, E. (2021). Auxiliary learning by implicit differentiation. In *International Conference on Learning Representations*.

[189] Nelder, J. A. and Mead, R. (1965). A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313.

[190] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*.

[191] Nguyen, T. Q. and Salazar, J. (2019). Transformers without tears: Improving the normalization of self-attention. In *Proceedings of the 16th International Conference on Spoken Language Translation*.

[192] Nickel, M. and Kiela, D. (2017). Poincaré embeddings for learning hierarchical representations. *Advances in neural information processing systems*, 30.

[193] Ojha, U., Singh, K. K., Hsieh, C.-J., and Lee, Y. J. (2020). Elastic-infogan: Unsupervised disentangled representation learning in class-imbalanced data. *Advances in neural information processing systems*, 33:18063–18075.

[194] Oono, K. and Suzuki, T. (2020a). Graph neural networks exponentially lose expressive power for node classification.

[195] Oono, K. and Suzuki, T. (2020b). Graph neural networks exponentially lose expressive power for node classification. In *International Conference on Learning Representations*.

[196] Papamichalis, M., Turnbull, K., Lunagomez, S., and Airoldi, E. (2022). Latent Space Network Modelling with Hyperbolic and Spherical Geometries. arXiv:2109.03343 [stat].

[197] Paredes, B. R., Argyriou, A., Berthouze, N., and Pontil, M. (2012). Exploiting unrelated tasks in multi-task learning. In *Artificial intelligence and statistics*, pages 951–959. PMLR.

[198] Park, Y.-H., Kwon, M., Choi, J., Jo, J., and Uh, Y. (2023). Understanding the latent space of diffusion models through the lens of riemannian geometry. *arXiv preprint arXiv:2307.12868*.

[199] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

[200] Pavllo, D., Feichtenhofer, C., Grangier, D., and Auli, M. (2019). 3d human pose estimation in video with temporal convolutions and semi-supervised training. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[201] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[202] Ramon, J. A. C., Herias, F. A. C., and Torres, F. (2011). Safe human-robot interaction based on dynamic sphere-swept line bounding volumes. *Robot. Comput. Integr. Manuf.*, 27(1):177–185.

[203] Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. (2016). Xnor-net: Imagenet classification using binary convolutional neural networks.

[204] Reynolds, W. F. (1993). Hyperbolic geometry on a hyperboloid. *The American Mathematical Monthly*, 100(5):442–455.

[205] Richemond, P. H., Tam, A., Tang, Y., Strub, F., Piot, B., and Hill, F. (2023). The edge of orthogonality: A simple view of what makes byol tick. In *International Conference on Machine Learning*, pages 29063–29081. PMLR.

[206] Rodriguez-Guerra, D., Sorrosal, G., Cabanes, I., and Calleja, C. (2021). Human-Robot Interaction Review: Challenges and Solutions for Modern Industrial Environments. *IEEE Access*, 9:108557–108578.

[207] Rosenberg, R. and Feigenson, L. (2013). Infants hierarchically organize memory representations. *Developmental science*, 16 4:610–21.

[208] Sala, F., De Sa, C., Gu, A., and Re, C. (2018). Representation tradeoffs for hyperbolic embeddings. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4460–4469. PMLR.

[209] Salehinejad, H., Sankar, S., Barfett, J., Colak, E., and Valaee, S. (2017). Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*.

[210] Sampieri, A., di Melendugno, G. M. D., Avogaro, A., Cunico, F., Setti, F., Skenderi, G., Cristani, M., and Galasso, F. (2022). Pose forecasting in industrial human-robot collaboration. In *European Conference on Computer Vision*, pages 51–69. Springer.

[211] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2008). The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80.

[212] Selsam, D., Lamm, M., Benedikt, B., Liang, P., de Moura, L., Dill, D. L., et al. (2018). Learning a sat solver from single-bit supervision. In *International Conference on Learning Representations*.

[213] Shah, J., Wiken, J., Breazeal, C., and Williams, B. (2011). Improved human-robot team performance using Chaski, a human-inspired plan execution system. *HRI 2011 - Proc. 6th ACM/IEEE Int. Conf. Human-Robot Interact.*, pages 29–36.

[214] Shazeer, N. and Stern, M. (2018). Adafactor: Adaptive learning rates with sublinear memory cost. In *Proceedings of the International Conference on Machine Learning (ICML)*.

[215] Shi, J., Zhang, H., and Li, J. (2019). Explainable and explicit visual reasoning over scene graphs.

[216] Shi, L., Wang, L., Long, C., Zhou, S., Zhou, M., Niu, Z., and Hua, G. (2021a). Sparse graph convolution network for pedestrian trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[217] Shi, Y., Huang, Z., Feng, S., Zhong, H., Wang, W., and Sun, Y. (2021b). Masked label prediction: Unified message passing model for semi-supervised classification. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

[218] Shimizu, R., Mukuta, Y., and Harada, T. (2021). Hyperbolic Neural Networks++. arXiv:2006.08210 [cs, stat].

[219] Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations (ICLR 2015)*.

[220] Simson, R. et al. (1838). *The elements of Euclid*. Desilver, Thomas.

[221] Singh, K. K., Ojha, U., and Lee, Y. J. (2019). Finegan: Unsupervised hierarchical disentanglement for fine-grained object generation and discovery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6490–6499.

[222] Skenderi, G., Capogrosso, L., Toaiari, A., Denitto, M., Fummi, F., Melzi, S., and Cristani, M. (2023a). Disentangled latent spaces facilitate data-driven auxiliary learning. *arXiv preprint arXiv:2310.09278*.

[223] Skenderi, G., Li, H., Tang, J., and Cristani, M. (2023b). Graph-level representation learning with joint-embedding predictive architectures. *arXiv preprint arXiv:2309.16014*.

[224] Sofianos, T., Sampieri, A., Franco, L., and Galasso, F. (2021a). Space-time-separable graph convolutional network for pose forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

[225] Sofianos, T., Sampieri, A., Franco, L., and Galasso, F. (2021b). Space-time-separable graph convolutional network for pose forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11209–11218.

[226] Speer, R., Chin, J., and Havasi, C. (2018). Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

[227] Speer, R. and Lowry-Duda, J. (2017). Conceptnet at semeval-2017 task 2: Extending word embeddings with multilingual relational knowledge. In *Proceedings of the International Workshop on Semantic Evaluation(SemEval)*.

[228] Standley, T., Zamir, A., Chen, D., Guibas, L., Malik, J., and Savarese, S. (2020). Which tasks should be learned together in multi-task learning? In *International Conference on Machine Learning*, pages 9120–9132. PMLR.

[229] Sun, F.-Y., Hoffman, J., Verma, V., and Tang, J. (2019). Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *International Conference on Learning Representations*.

[230] Suresh, S., Li, P., Hao, C., and Neville, J. (2021). Adversarial graph augmentation to improve graph contrastive learning. *Advances in Neural Information Processing Systems*, 34:15920–15933.

[231] Tan, Q., Liu, N., Huang, X., Choi, S.-H., Li, L., Chen, R., and Hu, X. (2023). S2gae: Self-supervised graph autoencoders are generalizable learners with graph masking. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pages 787–795.

[232] Thakoor, S., Tallec, C., Azar, M. G., Azabou, M., Dyer, E. L., Munos, R., Veličković, P., and Valko, M. (2021). Large-scale representation learning on graphs via bootstrapping. In *International Conference on Learning Representations*.

[233] Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., and Isola, P. (2020). What makes for good views for contrastive learning? *Advances in neural information processing systems*, 33:6827–6839.

[234] Topping, J., Di Giovanni, F., Chamberlain, B. P., Dong, X., and Bronstein, M. M. (2021). Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv preprint arXiv:2111.14522*.

[235] Torkar, C., Yahyanejad, S., Pichler, H., Hofbaur, M., and Rinner, B. (2019). Rnn-based human pose prediction for human-robot interaction. In *Proceedings of the ARW & OAGM Workshop 2019*, pages 76–80.

[236] Tu, H., Wang, C., and Zeng, W. (2020). Voxelpose: Towards multi-camera 3d human pose estimation in wild environment. In *European Conference on Computer Vision*, pages 197–212. Springer.

[237] van den Burg, G. and Williams, C. (2021). On memorization in probabilistic deep generative models. *Advances in Neural Information Processing Systems*, 34:27916–27928.

[238] Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

[239] Vandenhende, S., Georgoulis, S., and Van Gool, L. (2020). Mti-net: Multi-scale task interaction networks for multi-task learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16*, pages 527–543. Springer.

[240] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

[241] Veličković, P. (2023). Everything is connected: Graph neural networks. *Current Opinion in Structural Biology*, 79:102538.

[242] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2017). Graph attention networks. *arXiv preprint arXiv:1710.10903*.

[243] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph attention networks.

[244] Vianello, L., Mouret, J.-B., Dalin, E., Aubry, A., and Ivaldi, S. (2021). Human posture prediction during physical human-robot interaction. *IEEE Robotics and Automation Letters*.

[245] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.-A., and Bottou, L. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12).

[246] von Marcard, T., Henschel, R., Black, M., Rosenhahn, B., and Pons-Moll, G. (2018). Recovering accurate 3d human pose in the wild using imus and a moving camera. In *European Conference on Computer Vision (ECCV)*.

[247] Wald, J., Dhamo, H., Navab, N., and Tombari, F. (2020). Learning 3d semantic scene graphs from 3d indoor reconstructions.

[248] Wang, C., Wang, Y., Huang, Z., and Chen, Z. (2021). Simple baseline for single human motion forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 2260–2265.

[249] Wang, H. et al. (2003). Facial expression decomposition. In *Proceedings ninth IEEE international conference on computer vision*, pages 958–965. IEEE.

[250] Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., et al. (2020a). Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3349–3364.

[251] Wang, K., Lin, Y.-A., Weissmann, B., Savva, M., Chang, A. X., and Ritchie, D. (2019a). Planit: Planning and instantiating indoor scenes with relation graph and spatial prior networks. *ACM Transactions on Graphics (TOG)*, 38(4):132–147.

[252] Wang, K., Savva, M., Chang, A. X., and Ritchie, D. (2018). Deep convolutional priors for indoor scene synthesis. *ACM Transactions on Graphics (TOG)*, 37(4):1 – 14.

[253] Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., and Yu, P. S. (2019b). Heterogeneous graph attention network. In *Proceedings of The World Wide Web Conference (WWW)*.

[254] Wang, Y. (2015). Linear least squares localization in sensor networks. *EURASIP Journal on Wireless Communications and Networking*, 2015(1):1–7.

[255] Wang, Y., Giuliari, F., Berra, R., Castellini, A., Bue, A. D., Farinelli, A., Cristani, M., and Setti, F. (2020b). Pomp: Pomcp-based online motion planning for active visual search in indoor environments. In *Proceedings of the British Machine Vision Virtual Conference (BMVC)*.

[256] Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. (2019). Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR.

[257] Wu, S.-C., Wald, J., Tateno, K., Navab, N., and Tombari, F. (2021). Scenegraphfusion: Incremental 3d scene graph prediction from rgb-d sequences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7515–7525.

[258] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. (2020). A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24.

[259] Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. (2017). Aggregated residual transformations for deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5987–5995.

[260] Xie, Y., Xu, Z., and Ji, S. (2022a). Self-supervised representation learning via latent graph prediction. In *International Conference on Machine Learning*, pages 24460–24477. PMLR.

[261] Xie, Y., Xu, Z., Zhang, J., Wang, Z., and Ji, S. (2022b). Self-supervised learning of graph neural networks: A unified review. *IEEE transactions on pattern analysis and machine intelligence*, 45(2):2412–2429.

[262] Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L., and Liu, T. (2020). On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pages 10524–10533. PMLR.

[263] Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2018). How powerful are graph neural networks? In *International Conference on Learning Representations*.

[264] Xu, N., Liu, A.-A., Liu, J., Nie, W., and Su, Y. (2019). Scene graph captioner: Image captioning based on structural visual representation. *Journal of Visual Communication and Image Representation*, 58:477–485.

[265] Yang, X., Tang, K., Zhang, H., and Cai, J. (2019). Auto-encoding scene graphs for image captioning.

[266] Yang, X., Ye, J., and Wang, X. (2022). Factorizing knowledge in neural networks. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIV*, pages 73–91. Springer.

[267] Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., and Liu, T.-Y. (2021). Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34:28877–28888.

[268] Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., and Leskovec, J. (2018). Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31.

[269] You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., and Shen, Y. (2020). Graph contrastive learning with augmentations. *Advances in neural information processing systems*, 33:5812–5823.

[270] Yu, C., Ma, X., Ren, J., Zhao, H., and Yi, S. (2020). Spatio-temporal graph transformer networks for pedestrian trajectory prediction. In *The European Conference on Computer Vision (ECCV)*.

[271] Yu, T. and De Sa, C. M. (2021). Representing hyperbolic space accurately using multi-component floats. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 15570–15581. Curran Associates, Inc.

[272] Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. (2017). Deep sets. *Advances in neural information processing systems*, 30.

[273] Zamir, A. R., Sax, A., Shen, W., Guibas, L. J., Malik, J., and Savarese, S. (2018). Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3712–3722.

[274] Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*.

[275] Zhang, A., Lipton, Z. C., Li, M., and Smola, A. J. (2023). *Dive into deep learning*. Cambridge University Press.

[276] Zhang, B., Titov, I., and Sennrich, R. (2021a). Sparse attention with linear units. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

[277] Zhang, H., Wu, Q., Yan, J., Wipf, D., and Yu, P. S. (2021b). From canonical correlation analysis to self-supervised graph neural networks. *Advances in Neural Information Processing Systems*, 34:76–89.

[278] Zhang, J., Liu, H., Chang, Q., Wang, L., and Gao, R. X. (2020). Recurrent neural network for motion trajectory prediction in human-robot collaborative assembly. *CIRP annals*, 69(1):9–12.

[279] Zhao, W., Lopez, F., Riestenberg, M. J., Strube, M., Taha, D., and Trettel, S. (2023). Modeling Graphs Beyond Hyperbolic: Graph Neural Networks in Symmetric Positive Definite Matrices. In Koutra, D., Plant, C., Gomez Rodriguez, M., Baralis, E., and Bonchi, F., editors, *Machine Learning and Knowledge Discovery in Databases: Research Track*, Lecture Notes in Computer Science, pages 122–139, Cham. Springer Nature Switzerland.

[280] Zhao, Y. and Dou, Y. (2020). Pose-forecasting aided human video prediction with graph convolutional networks. *IEEE Access*, 8:147256–147264.

[281] Zheng, Y., Fan, J., Zhang, J., and Gao, X. (2019). Exploiting related and unrelated tasks for hierarchical metric learning and image classification. *IEEE Transactions on Image Processing*, 29:883–896.

[282] Zhou, D., Xiao, L., and Wu, M. (2011). Hierarchical classification via orthogonal transfer. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 801–808.

[283] Zhou, Q.-Y., Park, J., and Koltun, V. (2018). Open3D: A modern library for 3D data processing. *arXiv:1801.09847*.

[284] Zhou, Y., While, Z., and Kalogerakis, E. (2019). Scenegraphnet: Neural message passing for 3d indoor scene augmentation.