UNIVERSITY OF
CAMBRIDGE

Exploiting multimodality and structure in world representations

Cătălina Cangea

King's College

This dissertation is submitted for the degree of Doctor of Philosophy

March 2021

Declaration

This thesis is the result of my own work and includes nothing which is the outcome of work done in collaboration except as declared in the Preface and specified in the text. It is not substantially the same as any that I have submitted, or, is being concurrently submitted for a degree or diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. I further state that no substantial part of my thesis has already been submitted, or, is being concurrently submitted for any such degree, diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. It does not exceed the prescribed word limit for the relevant Degree Committee.

Cătălina Cangea
March, 2021

Abstract

Exploiting multimodality and structure in world representations

Cătălina Cangea

An essential aim of artificial intelligence research is to design agents that will eventually cooperate with humans within the real world. To this end, embodied learning is emerging as one of the most important efforts contributed by the machine learning community towards this goal. Recently developing sub-fields concern various aspects of such systems—visual reasoning, language representations, causal mechanisms, robustness to out-of-distribution inputs, to name only a few.

In particular, multimodal learning and language grounding are vital to achieving a strong understanding of the real world. Humans build internal representations via interacting with their environment, learning complex associations between visual, auditory and linguistic concepts. Since the world abounds with structure, graph-based encodings are also likely to be incorporated in reasoning and decision-making modules. Furthermore, these relational representations are rather symbolic in nature—providing advantages over other formats, such as raw pixels—and can encode various types of links (temporal, causal, spatial) which can be essential for understanding and acting in the real world.

This thesis presents three research works that study and develop likely aspects of future intelligent agents. The first contribution centers on vision-and-language learning, introducing a challenging embodied task that shifts the focus of an existing one to the visual reasoning problem. By extending popular visual question answering (VQA) paradigms, I also designed several models that were evaluated on the novel dataset. This produced initial performance estimates for environment understanding, through the lens of a more challenging VQA downstream task. The second work presents two ways of obtaining hierarchical representations of graph-structured data. These methods either scaled to much larger graphs than the ones processed by the best-performing method at the time, or incorporated theoretical properties via the use of topological

data analysis algorithms. Both approaches competed with contemporary state-of-the-art graph classification methods, even outside social domains in the second case, where the inductive bias was PageRank-driven. Finally, the third contribution delves further into relational learning, presenting a probabilistic treatment of graph representations in complex settings such as few-shot, multi-task learning and scarce-labelled data regimes. By adding relational inductive biases to neural processes, the resulting framework can model an entire distribution of functions which generate datasets with structure. This yielded significant performance gains, especially in the aforementioned complex scenarios, with semantically-accurate uncertainty estimates that drastically improved over the neural process baseline. This type of framework may eventually contribute to developing lifelong-learning systems, due to its ability to adapt to novel tasks and distributions.

The benchmark, methods and frameworks that I have devised during my doctoral studies suggest important future directions for embodied and graph representation learning research. These areas have increasingly proved their relevance to designing intelligent and collaborative agents, which we may interact with in the near future. By addressing several challenges in this problem space, my contributions therefore take a few steps towards building machine learning systems to be deployed in real-life settings.

Acknowledgements

Many claim their PhDs would not have been possible without the people mentioned in this section. I think mine would've still gone ahead no matter what, but in a very different and—dare I say—boring way! All the experiences I've had during these nearly four years (add to that the previous four, which set me on a research track) often diverge in nature from the somewhat controlled manner in which ideas appeared, work was done and pizzas were eaten while reviewing papers in my office, at the CL Cafe, in the Fishbowl, at conferences all over the world (well, US and Canada) or, more recently, on Zoom/gMeet/your favourite pandemic teleconferencing software... Still, the groups of people present in these two threads overlapped many times, and this half-journal-entry/half-list-of-top-N-awesome-people tells how they made my time in this degree more beautiful with every interaction we had.

I'll start at the end, by thanking my viva examiners—Nic Lane and Xavier Bresson—for reminding me through their insightful questions and future work suggestions that my research has been worthwhile and that it fits into the broader and richer picture, which I barely had space to outline in the thesis! Even though circumstances made me attend the viva from my room in my Romanian hometown(!), both of you made the virtual encounter seem like a vivid and easy-going in-person meeting—probably the dream of everyone who reaches thesis submission time.

I said my degree would've gone ahead no matter what, but there is one person who made it possible in the first place—my supervisor, Pietro Liò. His support went far beyond reviewing paper drafts and brainstorming sessions, turning into the one of an omnipresent friend-in-need and giving me better advice than most people I've ever met. I've told you this before, but will leave it in writing as well—I've been incredibly lucky to have you as a supervisor and will always appreciate you putting my happiness and sanity first, which is something not many PhD students seem to benefit from; thank you for believing in me, *'everything* [*seems to be going*] *OK'*!

The wonderful research group I was part of made these years much more enjoyable and social, be it during work or outside office hours—CL lunches and coffees, meetings on the

Backs, covid walks and hopefully many Italian-wine-infused formals to come! Felix and Indigo would always distract me from work in FE14, but lift my spirits in the meantime, with chats that became progressively rowing-focused, as Felix was persuaded to try it; Ben, Cris and Arian were humblingly smart collaborators, but also fun pub friends; Petar never stopped believing in me and my research output, even when I gave up; Paul was omnipresent in covid times for the 3pm coffee, when I'd be procrastinating, and kindly trusted me with his motorcycle gear; Helena inspired me with her enthusiasm for science and her fearless sports endeavours; Emma is a living proof that you can always take up something new, if you put your mind to it; Nikola never failed to join in on rants about various professional happenings, while sharing his extensive knowledge in paper feedback and asking difficult but important questions after AI talks; finally, Charlotte has been a great women@CL little sister and I'm happy for her upcoming PhD studentship—looking forward to BB coffees and Fitzbillies breakfasts with you!

As I was blessed with professional freedom during my PhD, I had the chance to be part of several communities, meeting amazing researchers who helped shape my knowledge of the field and explore worthwhile questions. At Mila, Aaron Courville offered me a fantastic perspective to embodied navigation, which is still my main thread of interest, almost three years after the internship! Eugene Belilovsky was of great help in the VideoNavQA project—I was excited to collaborate with him and Aaron in subsequent works led by Boris Knyazev, who is a model researcher to me, as I really admire his tenacity and rigor in paper-writing. At X, the moonshot factory, Qianyu Zhang was a great host and I was happy to be part of the team Olivia Hatalsky was leading towards future success! Working on real-world challenges is something that immensely motivates me, so I thank Jack Hidary for introducing us to X, and the early-stage project team I was part of in summer 2019 for the most motivating internship and close-knit collaboration I've ever done. Before and during my brief stint at Relation Therapeutics, Charlie Roberts and Jake Taylor-King made sure I shook hands with everyone in the office (it's OK, this was in February 2020) and shared their contagious excitement with me; Cristian Regep and Jyothish Soman were very helpful collaborators, making sure I'm getting the most out of the little time I was onboard. Finally, the last thing I did before writing my thesis was a four-and-a-half-month internship at DeepMind—here, Piotr Mirowski achieved the impossible, being the most helpful and present host an intern could ever hope for, in (yes, I will say it again) unprecedented times. The project I worked on in Raia Hadsell's team, getting immensely valuable feedback and brainstorming time with Piotr, her and Razvan Pascanu, unified many of my threads of interest and heavily contributed to my overarching research vision reflected by this thesis.

Supervising in my department is a two-way knowledge exchange. My undergraduate and research project students—Andy Wells, Felix Opolka, Aaron Solomon, Carlos Purves, Aaron Tjandra, Péter Mernyei, Mukul Rathi, Pavol Drotár—have taught me a great deal during our meetings and were a constant reminder of how this place is bursting with intelligence and enthusiasm. I wish you all the best in your careers and know for sure that you are capable of taking up any project, should you wish to.

The community in my College (King's, the PhD one, since I've been a member of three!) is incredibly friendly and down-to-earth. This includes the CS group, where Tim Griffin and Marwa Mahmoud were always lovely to me. It was a pleasure doing admissions interviews with them back in 2017, even though I had a terrible cold (apologies to any applicants who discovered they were ill back home...) I thank Tim for being wonderfully supportive ever since Part IB, even though I never managed to take up a Theory project, as evil Machine Learning got my attention instead. Marwa made the (gruesome, cold & wintery) third lockdown much sunnier on walks around town with hot chocolate—I very much hope to see you around King's in the future!

Once a rower, always annoying the non-rowers... this sport took entire months off my research time, but I will never regret all the flawless strokes King's W1 took on the water, the memories from boat club dinners and the friendships made in all crew lineups I've been part of! Rebekka was the greatest captain in non-covid times, because she never acted like her erg scores were better than the rest... although they were. Izzy was my extra-training buddy in Michaelmas 2019 and a constant inspiration—the only thing keeping you away from yet another erg/workout/circuit is yourself! Mary was the most terrifying yet excellent cox I've had so far and I will never forget how well she made us row, especially in Mays 2018. Amanda gets the best-captain-in-covid-times title, along with that of a kind and supportive friend—don't think many college clubs had this level of guidance from their captain in lockdowns and I thank her for keeping us sane with weekly workouts and leaderboards! Finally, Feli and Agnes have shown the world (or river Cam) that grads do it better, and I've been proud to do so alongside them.

There is no way I can cover all the ways in which my friends have made this degree worthwhile—my gratitude reaches far beyond a few lines of text. Towards Jess, my siamese rowing person across three different timezones, and Joy, my best friend on a *'Permanent Vacation'* in pandemic times. For Edgar, my ancient Tripos friend and the best housemate in the world, and Pavel, my Bayesian friend with whom I had the last breakfasts before covid and sang questionable Bon Jovi at karaoke. To Andreea, one of the kindest people and listeners I've ever met—let's have an unforgettable holiday this summer—and to Nathan and Jon, my amazing and loyal bandmates—let's keep playing music (yes, most of it is going to be about time) until this place goes underwater!

I generally don't have role models or view any people in my field as superstars, but if I were asked to choose a couple, I would start with my Computer Science teacher Daniela Lica, who inspired me to study algorithms in high school and so much more during university! This thesis is partly dedicated to you—thanks for being such a fundamental inspiration in my professional life; it must be wonderful being the name on so many people's minds, when they think of memorable figures who shaped their thoughts and early-career aspirations. I would also like to thank Marina Meilă for her support during my early research days and for the kind advice given during coffee breaks at ML conferences—I feel humbled and grateful to know you, such an accomplished and acknowledged researcher that I can only dream of becoming one day.

To my parents, Otilia and Emanuel Cangea—I will be eternally grateful for your unconditional love, appreciation and support, no matter how deviant some of my choices have been! Thank you for being such a wonderful unit, for believing in me, for raising me and for putting up with my nonsense and randomness, even in more recent times. This apple fell not far from the engineering tree.

Finally, with the risk of sounding like a broken record, I cannot ignore the changes that the (current...?!?) pandemic brought to me personally. Despite all the wreckage and disruption it caused globally, covid helped me turn around so many negative aspects of my life and figure out what makes me a better and happier person—certain habits, hobbies and wonderful people whose presence deeply affects me, regardless of how much time passes. My last year of the PhD was a highly inspirational one and I can only hope the change keeps blossoming.

"You can check out anytime you like
But you can never leave"
(The Eagles, Hotel California)

Contents

Glossary

BatchNorm Batch normalisation [89]. A method used for faster and more stable training of neural networks, which involves learning scaling and shift coefficients to multiply intermediate network representations by.

CNN Convolutional neural network [109]. A type of neural network that leverages spatial inductive biases and local connectivity, as opposed to full connectivity.

EQA Embodied Question Answering [38]. A VQA task from the perspective of an embodied agent, which needs to navigate in a house environment to find the answer.

FiLM Feature-wise linear modulation [145]. A method used for computing scale and shift coefficients for intermediate network representations (such as visual features), starting from a conditioning signal.

GNN Graph neural network [160]. A type of neural network that employs structural biases when processing the input data, thus incorporating the connectivity information present in the dataset.

LSTM Long short-term memory [81]. A type of neural network that leverages temporal biases in learning to represent the input signal, using several gating modules to enable selective 'forgetting'. Addresses the 'catastrophic forgetting' issue in vanilla recurrent neural networks.

MAC Memory, attention, and composition [88]. A MAC cell performs a single reasoning step over the current control state, conditioned by a separate signal (which can be language), by querying and updating the current memory state, typically consisting of visual features.

MP Message passing [61]. A paradigm for processing graph-structured data, which views the processing within a GNN as a series of operations that transmit information between nodes along the links connecting them to their neighbours.

NN Neural network [98]. A learnable function typically comprising a fixed number of fully-connected layers. Also known as an artificial neural network or a feedforward neural network.

NP Neural process [59]. A hybrid architecture that combines neural networks and Gaussian processes to obtain linear-time predictions with uncertainty estimates. This type of model is useful in challenging setups including multi-task, few-shot learning and low-data regimes.

SGD Stochastic gradient descent [157]. The most general approach to training neural networks, which takes small steps in parameter space, in the direction of the loss function gradient, over a mini-batch of examples.

TDA Topological Data Analysis [32]. A recent class of approaches designed for understanding the shape and other properties of datasets, using methods from algebraic and differential topology.

VQA Visual question answering [121]. A class of tasks where the machine learning system is given an image and a question as input and needs to produce an answer, either by classification (over a fixed number of answers) or via natural language generation.

Chapter 1

Introduction

The last decade of machine learning research has yielded extraordinary progress in sub-fields such as computer vision [30, 64, 105, 155, 158, 196], natural language processing [9, 21, 172, 180] and reinforcement learning [78, 79, 129, 130, 131, 132, 179, 204]. All of these developments have allowed machine learning systems to build increasingly powerful representations of specific knowledge about the real world or about artificial environments that simulate it. These methods have often been tailored to a single task, but other approaches—including ones used in meta-learning (or learning how to learn) [52, 150, 153, 193, 202], out-of-distribution generalisation [10, 80, 106, 171] and robustness-critical tasks [11, 67, 76, 189, 212]—enable adaptation to new settings or noisy and unpredictable ones. The latter kind of development has brought machine learning models one step closer to the challenging requirements that these systems will be faced with, as soon as they are deployed into the real-world.

What kind of competences would machine learning (ML) setups profit from the most? The forefront motivation of artificial intelligence research is to eventually integrate intelligent agents in humans' lives and enable them to co-operate [28, 50, 90] towards solving daily or more complex tasks. In order for this to work, we naturally aim to equip these intelligent systems with human-like capabilities: visual reasoning [7, 121, 205], language grounding [53, 182], understanding the functioning of an environment [40, 55, 116], collaborating with other agents and humans via language and actions—and lastly but not least importantly, navigating [125] within this space, all while handling the inherent uncertainty [60] about the environment.

Both humans and artificially-intelligent agents benefit from an egocentric perspective and experience of an environment [4, 68, 102, 146]. Therefore, research efforts in the embodied navigation, reasoning and learning sub-fields are an essential indicator of our progress towards the aforementioned goals—they help discover increasingly powerful

methods of building representations of the worlds that agents operate in. Numerous works [36, 40, 74, 84, 104, 119, 176, 197], along with proposed tasks [5, 33, 38, 65, 147, 149, 163, 175], often carried out in realistic environments [19, 44, 101, 159, 186, 191, 192], have been recently introduced at conferences and relevant workshop venues.

World representations have also been studied from a more general (and not necessarily embodied) perspective—for example, obtaining structured representations of scenes in reinforcement learning settings [63, 204], or directly from raw visual data [82, 96, 114, 148], and more complex approaches used to build causal representations [14, 23, 92, 171].

Reasonably, most of the tasks designed for embodied learning have not been 'solved' yet—several challenges remain to be addressed with regards to the points discussed above. Related sub-goals in this problem space also present various difficulties for current models in the research literature. To begin with, grounding linguistic concepts [53] in the visual and interactive cues received by an agent in an environment is such a complex task in itself, that an entire sub-community is working to improve on it. Research venues [2, 27, 56, 185] such as the ViGIL workshop [27], which I am co-organising in 2021, provide yearly opportunities for interdisciplinary communication on the topic between scientists from ML and other fields including psychology and linguistics—a suitable way to advance research in the field, given its highly heterogeneous nature.

Although a crucial step towards informed and effective world representations, integrating various capabilities and modules in an end-to-end embodied ML system is often highly challenging. Chapter 3 of the thesis tackles this issue, while also touching upon the language grounding one. As by-product of the study, another question arises: is designing an end-to-end trainable system always the best choice? Difficulty will also emerge from scaling to real-world environment size and complexity. For instance, navigating in cities [77, 113, 126] is not a trivial task—here, one cannot keep the entire navigation graph (wherein each node is a step taken by the agent) in working memory, as neighbourhoods in cities can quickly reach tens of thousands of nodes [127]. Instead, hierarchical techniques for graph summarisation may be useful in maintaining essential information at more manageable scales, by meaningfully compressing the raw navigation graph. Chapter 4 delves into this problem space and proposes two different techniques for graph coarsening that are applicable to any graphs with features living on their nodes. Finally, perhaps the most important challenge lies in handling out-of-distribution inputs [8], noisy data [49] and evolving distributions [69, 152]—all of which are likely to be present in the real-world settings agents will operate in. Chapter 5 presents a classification framework that is designed for challenging problem setups such as few-shot, multi-task and low availability of labelled data.

The rest of this chapter motivates and outlines a series of methods that I have designed and presented to the vision-and-language and graph representation learning communities. They each tackle a specific issue related to modules that intelligent agents might eventually require to operate in the real world—by either improving upon previous approaches or achieving comparable performance, whilst providing additional benefits.

1.1 Motivation and research questions

This thesis proposes a set of theoretical contributions in the form of machine learning algorithms, which produce representations of realistic environments and real-world information at an intermediate level within their processing. These representations are evaluated via several types of downstream tasks, ranging from question answering to individual sample and graph (or structured collection of samples) property prediction. In some cases, the manner in which evaluation tasks were constructed allowed me to draw conclusions about the explored machine learning paradigms. In others, by-products of the developed methods—such as increased interpretability or the presence of uncertainty estimates—were useful in highlighting the improved or complementary capabilities of these methods.

The eventual motivation of my research is for these types of modules to be incorporated in the internal system of an intelligent agent. All of them would be suitable in this setting, as they perform visual reasoning, processing of information in a hierarchical manner and property prediction with uncertainty estimates in challenging data scenarios (few-shot, multi-task, limited label availability), respectively. These components may aid the decision process of such an agent, while the latter explores the environment that it operates in, aggregating information over time and reasoning over the rich, multimodal and structure-informed memory (or knowledge base) it has thereby constructed.

Within all the tasks explored in this research, there still exists a considerable gap to human or perfect performance. It is thus essential to keep investigating the following research questions—perhaps in relation to other tasks or via more newly-proposed methods that have shown to improve performance on intermediate tasks (for example, extracting scene graphs from visual inputs). The central questions guiding the thesis are summarised below, along with the challenges encountered by previous research works.

1. *What is the level of performance that we can achieve in embodied reasoning tasks, if we obtain multimodal representations from visual-stream inputs, conditioned on the language signal from a question answering task?*
   Learning navigation, visual reasoning and language grounding for question answering has proved difficult when done within the same ML system, with language-

only baselines often surpassing these methods on the initial Embodied QA task. Chapter 3 presents VideoNavQA, a novel benchmark that views this task from a different standpoint—no navigation requirement and questions with higher variety and difficulty—along with methods that I developed for tackling the new task. VideoNavQA contains videos of embodied navigation in indoor environments and corresponding questions. The methods I designed are inspired by popular VQA paradigms and address the additional challenge of incorporating temporal processing in the visual pipeline. We provided initial results for this benchmark, which illustrate the relative effectiveness of VQA-based models against simple baselines and the considerable gap to human performance yet to be bridged.

2. *Can we obtain hierarchical representations of structured data that achieve state-of-the-art performance on downstream tasks, while sparsifying the mechanism that produces them? Or while making these representations more theoretically grounded?*
   Chapter 4 presents two different approaches to producing hierarchical representations of structured data, which were evaluated via graph classification performance on several benchmarks. The first method reduced the quadratic memory requirements which were necessary to achieve state-of-the-art performance at the time of the study—our pipeline achieved comparable results with linear requirements, thus scaling to bigger graphs. The second work addressed the lack of topological grounding in existing pooling layers, which would learn cluster assignment matrices end-to-end, strictly from the supervision signal. This would restrict the resulting assignments from being interpretable and preserving semantic and structural information at various scales. Results obtained by the pooling layer that I developed showed that adding a simple inductive bias in the node ranking mechanism—which is part of the cluster assignment process—achieves promising results, even on domains outside the ones initially expected.

3. *Can we model structured representations in a probabilistic framework that can better adapt and generalise to few-shot and multi-task contexts with changing data distributions?*
   Chapter 5 tackles the task of building representations of stochastic processes that generate structured data, leveraging the neural process framework. We have developed the Message Passing Neural Process by incorporating a relational inductive bias in its operation—the resulting framework performs classification by leveraging the structure present in datasets extracted from functions that are sampled from an underlying distribution. This resulted in significant improvements over neural processes across existing and novel tasks from several domains. We also showed, via qualitative and quantitative studies, that MPNPs produce uncertainty estimates for structured data that are more semantically-accurate and helpful in

active learning settings. The framework is highly suited to producing few-shot predictions and operating in multi-task settings, with only a few labelled samples, and adds a useful inductive bias to the neural process family.

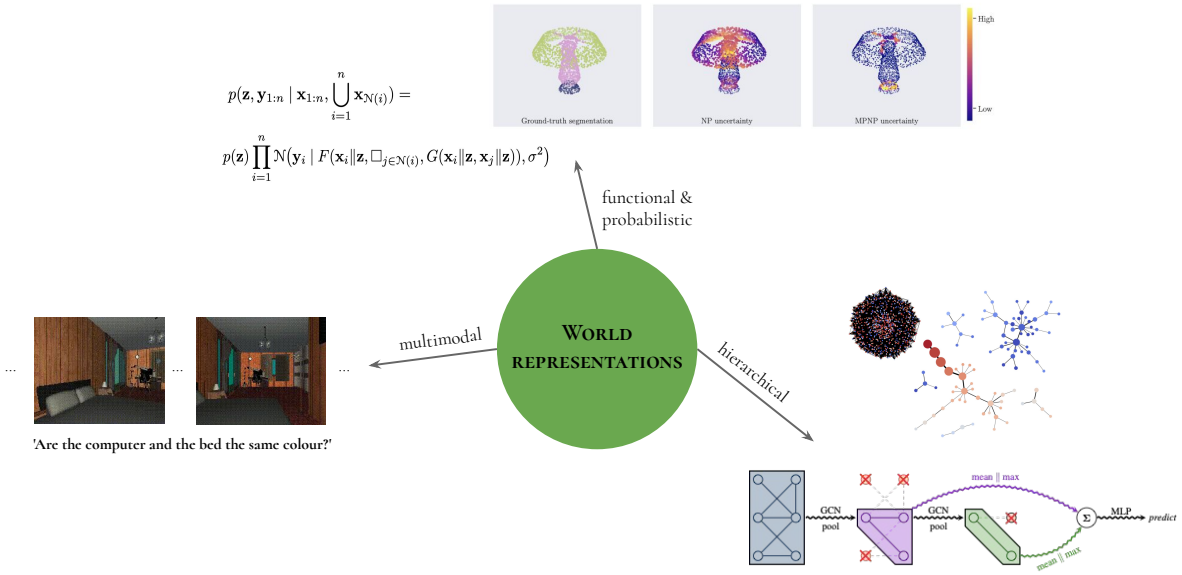

Figure 1.1: An illustration of the aspects explored in this thesis that can benefit world representations. (*Multimodal*:) The latter can be informed by visual streams of navigation inside an environment, where the ML system grounds the concepts from the linguistic signal (here, a question) in the visual information. (*Hierarchical*:) Representations can also be of this nature, since information exists at various scales and we might often need to choose one that is appropriate for the task that we are solving. (*Functional & probabilistic*:) A further step is to interpret the training data as samples across multiple functions, which were generated by the same stochastic process. Uncertainty estimates associated with predictions can be useful in a variety of real-world scenarios.

1.2 Thesis outline

The rest of this thesis is structured as detailed below. Contribution-wise, I first tackle the multimodal aspect of world representations, and follow with delving into the one that concerns hierarchical representations obtained via pooling methods. I conclude by introducing a framework for producing probabilistic representations of graph-structured data, which models the underlying stochastic generative process. Figure 1.1 illustrates the nature of these contributions.

Chapter 2—Background I start this chapter by reviewing the necessary theoretical background that was required to carry out the research presented in the contribution chapters—machine learning and neural network fundamentals, architectures with vari-

ous inductive biases (convolutional neural networks, long short-term memory, graph neural networks) and more complex frameworks (neural processes). I then move on to discussing each class of machine learning tasks tackled by the main research efforts, increasing in complexity whilst building up on previously discussed approaches.

Chapter 3—Multimodal learning for environment understanding Here, I describe how I created a novel benchmark for embodied question answering (EQA) tasks, then extended popular VQA paradigms to address the additional challenges introduced by this dataset. The aim was to obtain an idea of how well these models can perform on a more difficult variant of the initially-proposed EQA task, while removing the navigation requirement from the ML system. I developed VQA-style models for answering questions about visual navigation streams inside house environments. This essentially corresponds to tackling the EQA task from a different angle: QA performance is still being measured, but on restricted inputs and a much higher question difficulty. Our study reported results on the novel benchmark from single-modality baselines and video-and-language models. These results showed that multimodal methods indeed make use of the visual information, while enabling interesting conclusions about the effectiveness of VQA-style methods on temporal visual streams.

Chapter 4—Hierarchical representations of structured information In this chapter, I describe two different and perhaps complementary methods to performing graph coarsening within a graph classification pipeline. The first approach consists of a CNN-like architecture with sparse pooling layers—namely, modules which have a linear memory requirement in terms of the graph size. We showed that our model competed with the state-of-the-art method, DiffPool, on benchmark datasets; further, while the quadratic complexity of the DiffPool layer limited the maximum input size, our pipeline could accommodate significantly larger inputs. The second approach combines the expressiveness of graph neural networks with Mapper, a topological data analysis algorithm, to produce theoretically-grounded graph summaries. Specifically, I developed a pooling layer that operates by ranking node features using the PageRank function and then clusters them using Mapper. The graph classification performance was competitive with other contemporary state-of-the-art pooling approaches on standard benchmarks.

Chapter 5—Structural biases for probabilistic modelling in challenging scenarios This contribution involves adding structural inductive biases to the neural process framework, in order to exploit the structure available in the functions generated by stochastic processes. We leverage message passing within the encoder and decoder modules of the neural process, allowing the functional representation to be richer by incorporating structural information. Our Message Passing Neural Process (MPNP) framework was

evaluated on various data domains, with results validating the hypothesis that equipping NPs with the relational inductive bias yields better representations for classification. Furthermore, MPNPs were able to tackle few-shot and multi-task settings, where GNNs and other baselines failed to achieve better-than-chance results. Finally, the novel tasks designed by my collaborator showcase the applicability of MPNPs in a wide range of scenarios and propose important directions that the graph learning community can tackle in the near future.

Chapter 6—Conclusion and future directions In this final chapter, I summarise the contributions presented in the thesis and discuss future directions for each of them. Lastly, I present an overall vision for the purpose of learning world representations which comprise multimodal and structural information, both aspects having been explored within the research in my PhD. These representations will ideally be used by embodied agents during navigation, reasoning and task-solving in real-life(-like) environments.

1.3 Publications

The research efforts presented in this thesis have led to several publications, which I enumerate below. Most of them correspond to the main contributions in Chapters 3, 4 and 5, whilst the remaining ones are highly related in nature and aims or directly make use of these contributions:

1. **Cangea, C.***, Veličković, P.*, Jovanović, N., Kipf, T., & Liò, P. (2018). ***Towards Sparse Hierarchical Graph Classifiers***. arXiv preprint arXiv:1811.01287. Relational Representation Learning Workshop (NeurIPS 2018).

2. **Cangea, C.**, Belilovsky, E., Liò, P., & Courville, A. (2019). ***VideoNavQA: Bridging the Gap between Visual and Embodied Question Answering***. The 30th British Machine Vision Conference (BMVC 2019).

3. Mernyei, P., & **Cangea, C.** (2020). ***Wiki-CS: A Wikipedia-based Benchmark for Graph Neural Networks***. arXiv preprint arXiv:2007.02901. Graph Representation Learning and Beyond Workshop (ICML 2020).

4. Knyazev, B., de Vries, H., **Cangea, C.**, Taylor, G. W., Courville, A., & Belilovsky, E. (2020). ***Graph Density-Aware Losses for Novel Compositions in Scene Graph Generation***. The 31st British Machine Vision Conference (BMVC 2020).

5. Knyazev, B., de Vries, H., **Cangea, C.**, Taylor, G. W., Courville, A., & Belilovsky, E. (2020). ***Generative Graph Perturbations for Scene Graph Prediction***. arXiv preprint arXiv:2007.05756. Object-Oriented Learning Workshop (ICML 2020).

6. Bodnar, C.*, **Cangea, C.***, & Liò, P. (2020). ***Deep Graph Mapper: Seeing Graphs through the Neural Lens***. arXiv preprint arXiv:2002.03864. Topological Data Analysis and Beyond Workshop (NeurIPS 2020). Submitted to Frontiers in Big Data: Topology in Real-World Machine Learning and Data Analysis.

7. Day, B.*, **Cangea, C.***, Jamasb, A. R., & Liò, P. (2020). ***Message Passing Neural Processes***. arXiv preprint arXiv:2009.13895. Submitted to ICML 2021.

The 1st and 6th papers are joint-first authorship works which were presented at widely-attended NeurIPS workshops. They both introduce methods for producing hierarchical representations of graph-structured inputs. *'Towards Sparse Hierarchical Graph Classifiers'* has been widely acknowledged by the graph representation learning community, having received 70 citations at the time of submitting this thesis.

The 2nd publication represents the output of the collaboration started during my internship at Mila, with researchers Eugene Belilovsky and Aaron Courville, who was my supervisor. The work proposed a novel benchmark for measuring progress in embodied QA tasks and models that extended widely-used VQA paradigms. I presented this paper at the British Machine Vision Conference[1], one of the biggest computer vision research events, but also during a spotlight talk at the interdisciplinary ViGIL (Visually Grounded Interaction and Language) workshop at NeurIPS 2019[2]—which validated the relevance of this study to the research community.

The 7th study looked further into graph-structured data representations, by extending the neural process framework with a relational inductive bias. This contribution addresses more complex tasks which go beyond supervised learning, delving into multi-task and few-shot learning. We have submitted the manuscript to the Thirty-eighth International Conference on Machine Learning.

The 4th and 5th studies—conference and workshop papers, respectively—are related in nature to the vision-and-language study for embodied tasks that I carried out for the first publication. These two works, led by Boris Knyazev, focus on improving scene graph generation pipelines—namely, the prediction of node (object) and edge (relationship) classes from an input image. Their central aim was to ameliorate the generalisation issues that arise from training these systems on visual datasets with long-tailed data distributions. Scene graph generation systems have been increasingly studied lately and are thus an important part of the visual reasoning systems that embodied agents may end up using in practice.

[1] `https://bmvc2019.org/programme/detailed-programme/`
[2] `https://vigilworkshop.github.io/2019`

Finally, the 3rd work was led by Péter Mernyei, one of the students whom I supervised for their Part II (final-year undergraduate) dissertation project. He used the DGM graph summarisation technique proposed in the 3rd work to visualise Wiki-CS, the novel benchmark he devised for evaluating the performance of node classification architectures. Furthermore, DGM clearly illustrated differences in the structural and class distributions between Wiki-CS and the other, widely-used standard benchmarks (Cora, CiteSeer and PubMed). This manuscript was accepted to the Graph Representation Learning and Beyond Workshop at ICML 2020 and received a contributed talk slot, which confirms the importance of introducing a novel benchmark and illustrating differences in its data distribution from the current ones—a process in which DGM played an essential role.

Chapter 2

Background

This chapter starts by reviewing in Section 2.1 the machine learning theory applied in the contributions. Namely, Chapter 3 introduces multimodal approaches that use convolutional neural networks, long short-term memory and attention, Chapter 4 presents novel architectures comprising graph convolutional layers, and Chapter 5 describes a neural process framework with relational inductive biases in the general form of message passing. Section 2.2 then reviews the class of machine learning tasks tackled by each contribution, outlining potential challenges in every case. The purpose of this chapter is to make the thesis self-contained, but the material presented here is by no means exhaustive with respect to the corresponding research sub-fields.

2.1 Foundations, methods and building blocks

2.1.1 Feedforward neural networks

Feedforward neural networks (NNs), also known as multi-layer perceptrons (MLPs), are a class of parameterised machine learning models used to represent existing functions $f : X \to Y$. During learning, the parameters $\boldsymbol{\theta}$ of a neural network experience a sequence of updates, which produce the best possible approximation $\hat{f}_{\boldsymbol{\theta}}$ of a function f. Generally speaking, we wish to map an input $\boldsymbol{x} \in X$ to an output value $y \in Y$—the neural network computes $\hat{y} = \hat{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$. In classification settings, which are studied throughout this thesis, the quantity y represents the real label (or category) of the input $\boldsymbol{x}$ and $\hat{y}$ is the label predicted by the network.

Neural networks can be regarded as compositions of functions, such that:

$$\begin{aligned} f_{\boldsymbol{\theta}} &= f_{K\boldsymbol{\theta}_K} \circ \cdots \circ f_{1\boldsymbol{\theta}_1}, \\ \hat{y} &= f_{K\boldsymbol{\theta}_K}(\ldots f_{2\boldsymbol{\theta}_2}(f_{1\boldsymbol{\theta}_1}(\boldsymbol{x})))), \end{aligned} \tag{2.1}$$

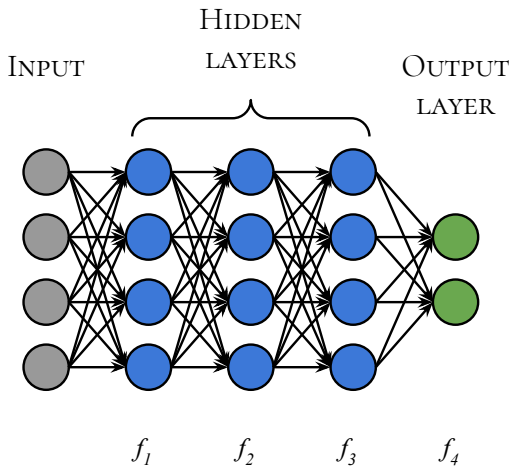

Figure 2.1: An example of a feedforward neural network with three hidden layers and a 2-D output layer. Each layer represents a separate function f_i—when composed, they yield the function f that the network is trained to represent.

where K is a given integer with $K \geq 1$ and $\boldsymbol{\theta} = \bigcup_{i=1}^{K} \boldsymbol{\theta}_i$. Every $f_{i\boldsymbol{\theta}_i}$ corresponds to a layer in the network with associated parameters $\boldsymbol{\theta}_i$[1]. A layer receives as inputs the outputs of the previous layer, as per Equation 2.1. The simplest type of neural network layer is the fully-connected one, which computes a linear transformation of its input $\boldsymbol{h}_i$:

$$\boldsymbol{h}_{i+1} = \boldsymbol{W}_i \boldsymbol{h}_i + \boldsymbol{b}_i, \tag{2.2}$$

with $\boldsymbol{\theta}_i = \boldsymbol{W}_i \cup \boldsymbol{b}_i$. The outputs of a fully-connected layer are usually passed through a point-wise activation function, which intuitively signals the relative strength of certain patterns in the input features detected by the layer. The most popular activation function is the rectified linear unit (ReLU [62]):

$$\text{ReLU}(\boldsymbol{x}) = \max(\boldsymbol{0}, \boldsymbol{x}). \tag{2.3}$$

The output layer of a neural network can categorise inputs via the softmax function:

$$o_i = \text{softmax}(\boldsymbol{h})_i = \frac{\exp(\boldsymbol{h}_i)}{\sum_{c=1}^{C} \exp(\boldsymbol{h}_c)}, \tag{2.4}$$

where C is the number of categories present in the dataset. The resulting output is a valid probability distribution, with the predicted class being output as $\hat{y} = \text{argmax}_i \, o_i$.

[1]Note that $\boldsymbol{\theta}_i$ can be an empty set, in the case of activation and output layers.

2.1.2 Model optimisation

Neural networks are typically trained using maximum likelihood—in a classification setup, the learning procedure aims to minimise a negative log-likelihood cost function J that encodes the cross-entropy between the true, underlying data distribution, p_{data}, and the distribution modelled by the network, p_{model}:

$$\begin{aligned} J(\boldsymbol{\theta}) &= -\mathbb{E}_{\boldsymbol{x},y\sim p_{\text{data}}} \log p_{\text{model}}(y \mid \boldsymbol{x}) \\ &= -\sum_{c=1}^{C} y_c \log \hat{y}_c. \end{aligned} \tag{2.5}$$

The cost function can be further expressed as $\mathbb{E}_{\boldsymbol{x},y\sim\hat{p}_{\text{data}}}\mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{x}), y)$, where $\hat{p}_{\text{data}}$ is the empirical data distribution and $\mathcal{L}$ is the cross-entropy loss function in Equation 2.5. We aim to reduce the expected generalisation error, which is given by the objective function over the real data distribution: $J^*(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x},y\sim p_{\text{data}}}\mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{x}), y)$. To achieve this, we minimise the empirical risk over the training dataset containing m samples:

$$\mathbb{E}_{\boldsymbol{x},y\sim\hat{p}_{\text{data}}}\mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{x}), y) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i). \tag{2.6}$$

The learning process updates the parameters of the network by taking small steps in the direction of the loss gradient $\hat{\boldsymbol{g}}$ over a mini-batch of b examples from the training set:

$$\begin{aligned} \hat{\boldsymbol{g}} &= \frac{1}{b}\nabla_{\boldsymbol{\theta}}\sum_{i=1}^{b}\mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i), \\ \boldsymbol{\theta} &= \boldsymbol{\theta} - \alpha\hat{\boldsymbol{g}}. \end{aligned} \tag{2.7}$$

This approach is called stochastic gradient descent (SGD) [157] and optimises the network parameters using unbiased estimates $\hat{\boldsymbol{g}}$ of the exact gradient of the generalisation error. An entire pass through the training set is called an epoch, with training typically spanning multiple epochs. The number of epochs can vary, depending on the size of the dataset, the number of model parameters, the type of optimisation algorithm used and the learning rate α employed.

Building upon SGD, the Adam optimisation algorithm [95] is widely used in practice and for the experiments described in this thesis. Adam adapts the learning rate to individual parameters at each step t, using estimates of the first ($\boldsymbol{m}_t$) and second ($\boldsymbol{v}_t$)

order moments of the gradients:

$$\begin{aligned}
\boldsymbol{m}_t &= \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t, \\
\boldsymbol{v}_t &= \beta_2 \boldsymbol{v}_{t-1} + (1-\beta_2)\boldsymbol{g}_t^2, \\
\hat{\boldsymbol{m}}_t &= \boldsymbol{m}_t(1-\beta_1^t), \\
\hat{\boldsymbol{v}}_t &= \boldsymbol{v}_t(1-\beta_2^t), \\
\boldsymbol{\theta}_t &= \boldsymbol{\theta}_{t-1} - \alpha\hat{\boldsymbol{m}}_t(\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon).
\end{aligned} \tag{2.8}$$

2.1.3 Regularisation

2.1.3.1 Dropout

The purpose of dropout [169] is to regularise a neural network model by approximating training and evaluating over an ensemble of networks, via the (exponentially-many) sub-networks of the original model. The approach relies on randomly masking outputs of intermediate layers—these are no longer forwarded to subsequent layers and thus do not contribute to the final network prediction.

If $\boldsymbol{\mu}$ is a mask vector over all intermediate outputs in the network, then the resulting sub-network produces a probability distribution $p(y \mid \boldsymbol{x}, \boldsymbol{\mu})$—the final distribution $p_{\text{ensemble}}(y \mid \boldsymbol{x})$ is computed by taking the geometric mean over a number of sampled masks and normalising over all possible outputs (in the classification scenario, these correspond to categories), where d is the number of units that may be dropped:

$$\begin{aligned}
\tilde{p}_{\text{ensemble}}(y \mid \boldsymbol{x}) &= \sqrt[2^d]{\prod_{\boldsymbol{\mu}_i} p(y \mid \boldsymbol{x}, \boldsymbol{\mu}_i)}, \\
p_{\text{ensemble}}(y \mid \boldsymbol{x}) &= \frac{\tilde{p}_{\text{ensemble}}(y \mid \boldsymbol{x})}{\sum_{y'} \tilde{p}_{\text{ensemble}}(y' \mid \boldsymbol{x})}.
\end{aligned} \tag{2.9}$$

2.1.3.2 Batch normalisation

One way to train networks faster is to ensure that the inputs to all layers have similar magnitudes, which is the key idea behind batch normalisation (BatchNorm) [89]. This way, a learning rate applied across all network layers will yield consistent updates, eliminating the burden of carefully selecting an initialisation distribution for each layer.

For a given activation map across a mini-batch, $\boldsymbol{H} \in \mathbb{R}^{b \times f}$, where b is the mini-batch size and f is the feature vector dimensionality, BatchNorm rescales $\boldsymbol{H}$ using the mean $\boldsymbol{\mu}$ and

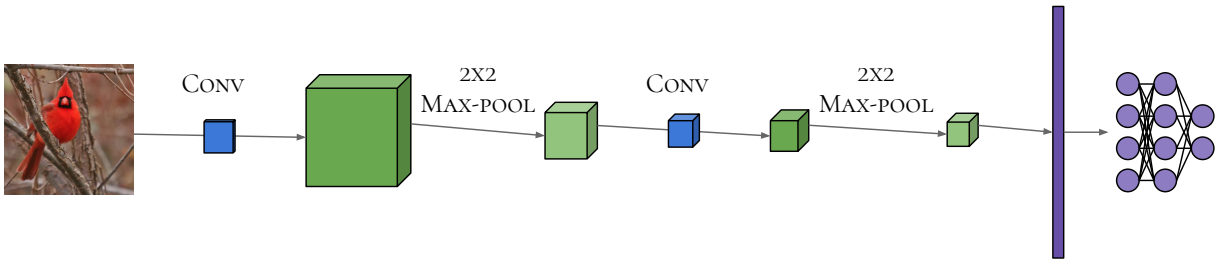

Figure 2.2: An illustration of a CNN with 2 convolutional layers and corresponding max-pooling steps, followed by a flattening operation and two fully-connected layers. *Blue:* convolutional kernels with parameters; *green:* visual features; *purple:* summarised features and prediction pipeline.

standard deviation $\boldsymbol{\sigma}$ of each unit in the layer (where $\boldsymbol{h}_i = (\boldsymbol{H})_i$):

$$\begin{aligned} \boldsymbol{H}' &= \frac{\boldsymbol{H} - \boldsymbol{\mu}}{\boldsymbol{\sigma}}, \\ \boldsymbol{\mu} &= \frac{1}{m}\sum_{i=1}^{b} \boldsymbol{h}_i, \\ \boldsymbol{\mu} &= \sqrt{\delta + \frac{1}{m}\sum_{i=1}^{b}(\boldsymbol{h}_i - \boldsymbol{\mu})^2}. \end{aligned} \tag{2.10}$$

The original study even showed that BatchNorm removed the need for dropout in contemporary state-of-the-art networks, thus producing a strong regularising effect.

2.1.4 Convolutional neural networks

This class of models, often referred to as CNNs [110], has been designed to incorporate additional biases in neural network processing. The convolutional architectures described in this thesis contain 2-D and 3-D convolutional layers—these leverage spatial biases, allowing the regular, grid-like structure in visual data (where each pixel has a fixed number of neighbours with fixed relative displacements) to be exploited.

A typical CNN model takes visual tensors as input and contains a sequence of alternating convolutional and pooling steps. The convolutional layers output feature maps, which are then downsampled by the pooling operations—an increasingly compact representation of the input is thus constructed in this part of the network. A summarisation layer is then applied, with the resulting feature vector being fed to an MLP that outputs predictions required by the downstream task. Figure 2.2 shows an example of a CNN.

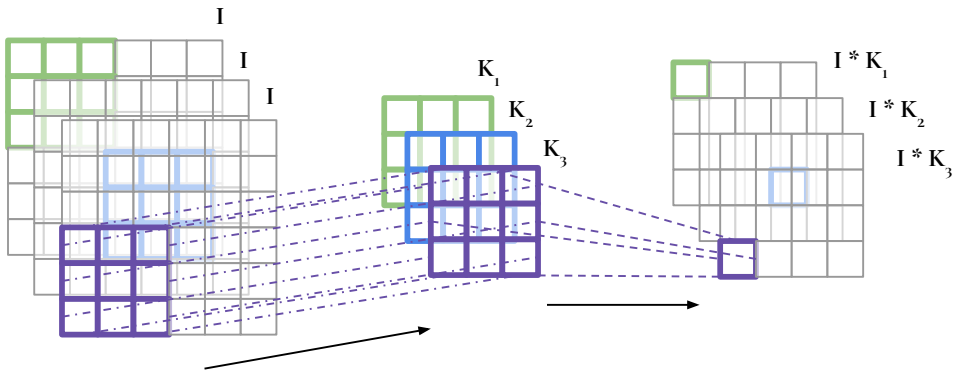

Figure 2.3: An illustration of a convolutional layer with 3 kernels that produces 3 corresponding feature maps, by sliding each kernel across the input.

2.1.4.1 Convolutional layers

A convolutional layer takes as input a multidimensional tensor that contains visual features (or, in the case of the first network layer, the raw input image). The tensor is passed through a set of kernels, each of these computing its own feature map of the input. For a 2-D tensor $\boldsymbol{I}$ and a kernel $\boldsymbol{K}$, the convolution operation is:

$$\begin{aligned}(\boldsymbol{I} * \boldsymbol{K})_{ij} &= \sum_h \sum_w I_{hw} K_{i-h,j-w} \\ &= \sum_h \sum_w I_{i-h,j-w} K_{h,w},\end{aligned} \tag{2.11}$$

but most machine learning frameworks will implement the following form, which does not flip the kernel and is called cross-correlation, depicted in Figure 2.3:

$$(\boldsymbol{I} * \boldsymbol{K})_{ij} = \sum_h \sum_w I_{i+h,j+w} K_{h,w}. \tag{2.12}$$

Each kernel outputs a feature map that indicates at which input location the corresponding feature has been detected. This approach has a crucial advantage over fully-connected layers: kernels contain far fewer parameters than the input and reuse these parameters across the input (in contrast to connecting each unit to every input location). This, in turn, makes a convolutional layer translation-equivariant—a feature will be detected regardless of its location in the input. In the case of 3-D input tensors, both spatial and temporal biases are leveraged. Intuitively, the kernel detects features and their relative displacement across a few timesteps:

$$(\boldsymbol{I} * \boldsymbol{K})_{ijk} = \sum_t \sum_h \sum_w I_{i+t,j+h,k+w} K_{t,h,w}. \tag{2.13}$$

Figure 2.4: Two images of birds from the Caltech-UCSD Birds 200 dataset[2].

2.1.4.2 Pooling layers

The pooling operation follows the feature detection stage and has the purpose of compressing the visual representation for further processing. This is achieved by computing summary statistics for each input location, within a (typically) square-grid neighbourhood. CNNs often use 2×2 2-D max-pooling with a stride of $(2, 2)$. Namely, each non-overlapping 2×2 patch in the input corresponds to a single location in the output feature map, which contains the maximum value across the patch.

In addition to allowing the network to be more memory-efficient with respect to the input size, this process also enforces a translation-invariant representation, where small shifts in the inputs do not affect the resulting feature map. To illustrate the idea, if a bird is present in two images as in Figure 2.4, the corresponding output representations will still encode the fact that a bird is found at the centre of each image.

2.1.5 Long short-term memory

Another type of inductive bias is incorporated in deep learning architectures via recurrent neural networks (RNNs). These models process sequential data by exploiting the temporal dependencies present in this domain. For an input $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$ and network state $\boldsymbol{h}_t$ at time t, the general mathematical form for expressing RNN computations is:

$$\boldsymbol{h}_t = f_{\boldsymbol{\theta}}(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t). \tag{2.14}$$

Intuitively, the state of the network $\boldsymbol{h}_t$ selectively encodes the most relevant aspects of the input sequence detected so far, which are useful for the downstream task. However,

[2] `http://www.vision.caltech.edu/visipedia/CUB-200.html`

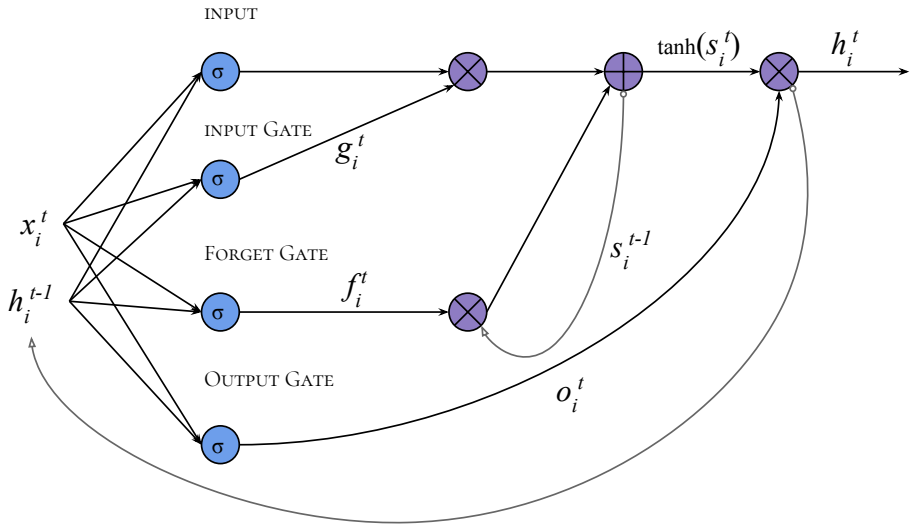

Figure 2.5: A graphical description of the LSTM cell.

backpropagating through the entire input sequence gives rise to vanishing gradients, due to multiplying increasingly-many gradient values which are usually smaller than 1 (in the rare opposite case, exploding gradients are produced).

The long short-term memory (LSTM) [81] model is a type of RNN that addresses this problem, by introducing gating mechanisms with non-vanishing/exploding derivatives. The key aspect here is that the hidden state contains a self-loop which is weighted by a forget gate unit—the latter produces outputs between 0 and 1 and allows the input sequence to condition the flow of information across time. The computation of the i-th LSTM cell is illustrated in Figure 2.5 and summarised by the following equations:

$$
\begin{aligned}
f_i^t &= \sigma\left(b_i^f + \sum_j U_{ij}^f x_j^t + \sum_j W_{ij}^f h_j^{t-1}\right) \\
g_i^t &= \sigma\left(b_i^g + \sum_j U_{ij}^g x_j^t + \sum_j W_{ij}^g h_j^{t-1}\right) \\
s_i^t &= f_i^t s_i^{t-1} + g_i^t \sigma\left(b^s + \sum_j U_{ij}^s x_j^t + \sum_j W_{ij}^s h_j^{t-1}\right) \\
o_i^t &= \sigma\left(b_i^o + \sum_j U_{ij}^o x_j^t + \sum_j W_{ij}^o h_j^{t-1}\right) \\
h_i^t &= \tanh\left(s_i^t\right) o_i^t,
\end{aligned}
\tag{2.15}
$$

where h_i^t is the output of the cell at time t, $\boldsymbol{b}^*$, $\boldsymbol{U}^*$ and $\boldsymbol{W}^*$ are biases, input weights and recurrent weights, respectively, for all corresponding cell modules (f—forget gate, s—cell state, g—input gate, o—output gate). All gates use sigmoid (σ) activations.

2.1.6 Attention

Initially proposed for neural machine translation [9], the attention mechanism is an encoder-decoder model that takes as input a sequence $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$ and outputs another sequence $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{T'}\}$. The i-th token of the latter encompasses information about the previous output tokens $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{i-1}\}$ and weighted information about all the input tokens. Namely, the model attends over the input elements, deciding for each new sequence which parts of it contain more relevant information to produce the i-th output.

One might note that this approach is applicable beyond machine translation, in any task where certain regions of the input sequence are more informative than others.

The input sequence is first passed through an encoder, which can either be a bidirectional RNN or another type of model that outputs a corresponding $\boldsymbol{h}_t$ for each $t \in \{1, \ldots, T\}$. The decoding stage then models the conditional probability of the i-th output, depending on the previous outputs and all inputs:

$$p(\boldsymbol{y}_i \mid \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{i-1}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T) = g(\boldsymbol{y}_{i-1}, \boldsymbol{s}_i, \boldsymbol{c}_i). \tag{2.16}$$

In Equation 2.16, $\boldsymbol{s}_i$ is the hidden state of an RNN f at time i and $\boldsymbol{c}_i$ is a context vector that contains weighted information about the encoded inputs $\boldsymbol{h}_i$:

$$\begin{aligned} \boldsymbol{s}_i &= f(\boldsymbol{s}_{i-1}, \boldsymbol{y}_{i-1}, \boldsymbol{c}_i) \\ \boldsymbol{c}_i &= \sum_{j=1}^{T} \alpha_{ij} \boldsymbol{h}_j. \end{aligned} \tag{2.17}$$

The attention weights α_{ij} are computed by an alignment module a, which indicates how strongly the j-th input and i-th output match. The module is typically parameterised by an MLP that is incorporated and trained end-to-end with the rest of the architecture.

$$\begin{aligned} \alpha_{ij} &= \frac{\exp(e_{ij})}{\sum_{k=1}^{T} \exp(e_{ik})} \\ e_{ij} &= a(\boldsymbol{s}_{i-1}, \boldsymbol{h}_j). \end{aligned} \tag{2.18}$$

Intuitively, the weights α_{ij} signal to the RNN the importance of each $\boldsymbol{h}_j$ from the encoded sequence for generating the next hidden state $\boldsymbol{s}_i$ and, in turn, the next output $\boldsymbol{y}_i$.

2.1.7 Message passing and graph neural networks

So far, this chapter has reviewed various changes that feedforward neural networks have undergone to incorporate inductive biases. CNNs thereby perform spatially-

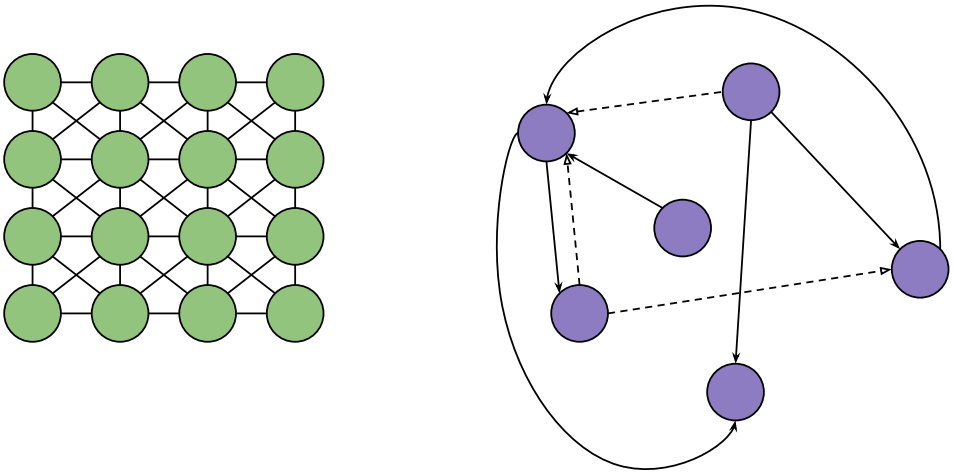

Figure 2.6: An example of the arbitrary connectivity that can be encountered in graph-structured data (*right*), as opposed to the visual inputs (*left*) typically processed by convolutional neural networks. In the former case, links can have different types and directions, whereas grid-like data has a fixed, non-directional connectivity pattern.

aware processing, whereas LSTMs and attention[3] are readily applicable to sequential data. However, much of the data used to solve contemporary real-world tasks—traffic prediction, drug discovery or social network recommendation, to name only a very few—has irregular intra-dependencies of various types. This heavily contrasts with the grid-like connections found in visual data or the ones dictated by temporal structure in time series and language data.

Figure 2.6 illustrates the relative complexity of graph-structured inputs, in relation to the one of visual data commonly processed by CNNs. Perhaps most importantly, a CNN-style kernel cannot be readily applied to a graph-structured input, since each node might have a different number of neighbours and it is almost impossible to make assumptions about this quantity in a real-world scenario—a more flexible operator is thus required for training neural networks on graphs.

A notable development towards addressing this problem is the graph convolutional network (GCN) layer [97], which generalises the previously-introduced convolutional layer to handle graph-structured inputs. Assume a given graph $G = (\boldsymbol{H}^i, \boldsymbol{A})$, where $\boldsymbol{H}^i \in \mathbb{R}^{n \times d_i}$ are intermediate node features of dimensionality d_i and $\boldsymbol{A}$ is the adjacency

[3]Note that other forms of attention have also been proposed, including the widely-used Transformer attention that leverages the set bias without assuming any ordering of the input elements. However, I have only used recurrent attention in the work, which was described previously.

matrix of the graph. The GCN layer models the following propagation rule:

$$\begin{aligned} \boldsymbol{H}^{i+1} &= \text{GCN}(\boldsymbol{H}^i, \boldsymbol{A}) \\ &= \sigma\left(\hat{\boldsymbol{D}}^{-\frac{1}{2}}\hat{\boldsymbol{A}}\hat{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{H}^i\boldsymbol{W}^i\right), \end{aligned} \tag{2.19}$$

where $\boldsymbol{W} \in \mathbb{R}^{d_i \times d_{i+1}}$ is a learnable weight matrix for the current layer and $\boldsymbol{H}^{i+1} \in \mathbb{R}^{n \times d_{i+1}}$. Equation 2.19 is essentially an approximation of a spectral convolution that is far less expensive when scaling to larger graphs. Note that the layer processing involves adding self-loops to the adjacency matrix—$\hat{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}_n$—and the use of an additional matrix $\hat{\boldsymbol{D}}$—the diagonal node degree matrix of $\hat{\boldsymbol{A}}$, with $\hat{D}_{ii} = \sum_j \hat{A}_{ij}$. These changes allow the factor $\hat{\boldsymbol{D}}^{-\frac{1}{2}}\hat{\boldsymbol{A}}\hat{\boldsymbol{D}}^{-\frac{1}{2}}$ to replace $\boldsymbol{I}_n - \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$ in the initial approximation of the spectral convolution:

$$g_\theta * x \approx \theta\left(\boldsymbol{I}_n - \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}\right)x, \tag{2.20}$$

where x is a single-channel input signal and $g_\theta = \text{diag}(\theta)$ is a filter with $\theta \in \mathbb{R}^n$. This approximation can be naturally generalised to a multi-channel input signal $\boldsymbol{X} \in \mathbb{R}^{n \times c}$ as $\boldsymbol{X}' = \hat{\boldsymbol{D}}^{-\frac{1}{2}}\hat{\boldsymbol{A}}\hat{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{X}\boldsymbol{\Theta}$, where $\boldsymbol{\Theta} \in \mathbb{R}^{c \times f}$ contains filter parameters for each of the c channels and $\boldsymbol{X}' \in \mathbb{R}^{n \times f}$ is the convolved signal. This is the form that yields Equation 2.19.

Analysing the GCN propagation rule, we note that each node is updated based on its current features and the ones of its neighbours, since the adjacency matrix is part of the multiplication. Additionally, the self-loops added to $\boldsymbol{A}$ enforce the current node features to be preserved across updates; certain variants of GCN layers go one step further and use $\hat{\boldsymbol{A}} = \boldsymbol{A} + 2\boldsymbol{I}_n$, for node representations with a stronger egocentric focus.

This perspective on GCNs paves the way for a discussion of an even more general paradigm—the message passing (MP) neural network model [61], which subsumes many graph neural network (GNN) models. An MP layer takes as input the features of all nodes after p message passing steps $\boldsymbol{H}^p \in \mathbb{R}^{n \times d'}$, where d' is the number of node features, the adjacency matrix $\boldsymbol{A}$ and the corresponding collection of k-dimensional edge features[4] $\{\boldsymbol{e}_{ij} \in \mathbb{R}^k | A_{ij} = 1\}$. The MP step then updates the nodes as follows:

$$\begin{aligned} \boldsymbol{h}_i^{p+1} &= \text{MP}(\boldsymbol{H}^p, \boldsymbol{A}) \\ &= F(\boldsymbol{h}_i^p, \bigoplus_{j \in \mathcal{N}(i)}, G(\boldsymbol{h}_i^p, \boldsymbol{h}_j^p, \boldsymbol{e}_{ij})), \end{aligned} \tag{2.21}$$

where F and G are learnable functions, $\mathcal{N}(i) = \{j \mid A_{ij} = 1\}$ and $\bigoplus$ is a permutation-invariant aggregation function. The function G can be viewed as the message encoder, since it embeds, for each edge, the tuple containing its features and the endpoint node

[4] Although not present in the experimental setups in this thesis, edge features are also incorporated in the MP formulation.

features. Following the computation of all message embeddings, the aggregator $\bigoplus$ is applied to produce an overall neighbourhood embedding, which gets passed to the final function F, along with the current node features. In practice, F and G are often MLPs, with optional activation functions following their application.

2.1.8 Neural processes

Neural Processes (NPs) [59] are a class of models that combine the strengths of neural networks and Gaussian processes. They learn to represent a stochastic process using labelled samples from its instantiations f_i, with a global latent variable z modelling the stochasticity of the learned functions. At test time, only a few labelled points from the current dataset (namely, the function f_i) are required to produce predictions for the rest of the points, along with associated uncertainties, in linear time.

To summarise the task, the NP is given a set of points with features $\boldsymbol{X}$, partially labelled by a function $f : X \to Y$ that has been sampled from a distribution over functions $\mathcal{D}$. The NP then predicts labels for a subset of the unlabelled points.

To achieve this, the NP is trained on a set of functions sampled from $\mathcal{D}$ and tested on a disjoint set of functions from the same distribution. For each function f_i, a dataset contains tuples (x_j, y_j), where $y_j = f_i(x_j)$. Their joint probability distribution can be written as $p(y_{1:n}|x_{1:n}) = \int p(f_i)p(y_{1:n}|f_i, x_{1:n})df_i$. Assuming observation noise $Y_j \sim \mathcal{N}(f_i(x_j), \sigma^2)$ and a neural network γ modelling the stochastic process instance f_i (that is, $\gamma(x, z) = f_i(x)$, where z is a random vector that mimics the randomness of f_i), we obtain the generative model:

$$p(z, y_{1:n}|x_{1:n}) = p(z)\prod_{j=1}^{n}\mathcal{N}(y_j|\gamma(x_j, z), \sigma^2), \tag{2.22}$$

where $p(z)$ is a multivariate normal distribution. Learning the non-linear function γ requires amortised variational inference on the evidence lower bound (ELBO), using a neural-network-parameterised posterior $q(z|x_{1:n}, y_{1:n})$. Model generation starts with the NP receiving a set of m context points $\mathcal{C} = \{(x_j, y_j)\}_{j=1}^{m}$ sampled from f_i. The model then predicts the values $y_j = f_i(x_j)$ for n target points $\mathcal{T} = \{x_j\}_{j=1}^{n}$; namely, the m original context points and $m - n$ previously unseen target points. To match this setup, we further isolate the context set $x_{1:m}, y_{1:m}$ from the target set $x_{m+1:n}, y_{m+1:n}$ in Equation 2.22. The final variational approximation to the lower bound objective can be expressed as:

$$\log p(y_{m+1:n}|x_{1:n}, y_{1:m}) \geq \mathbb{E}_{q(z|x_{1:n},y_{1:n})}\Big[\sum_{j=m+1}^{n}\log p(y_j|z, x_j) + \log\frac{q(z|x_{1:m}, y_{1:m})}{q(z|x_{1:n}, y_{1:n})}\Big]. \tag{2.23}$$

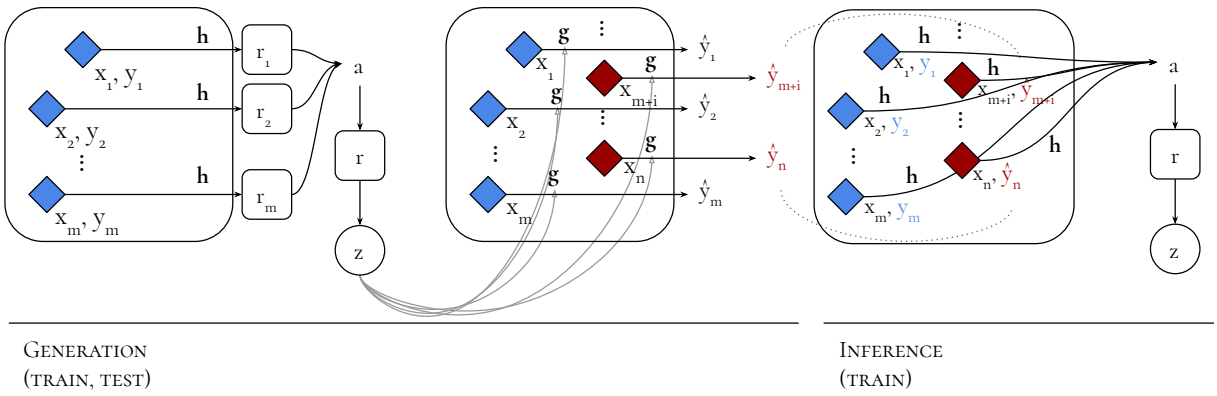

Figure 2.7: The computational graph of the Neural Process model. (*Generation:*) The encoder **h** produces a representation r_i for each (labelled) context point (x_i, y_i). These are passed through the aggregator function a which yields a summary r. This, in turn, parameterises the global latent variable z. The latter is sampled to condition the decoding process. Here, the decoder **g** processes the sample along with every (unlabelled) target point to predict a corresponding label $\hat{y}_i$, along with an associated uncertainty. (*Inference:*) The global latent posterior update is achieved by encoding the entire target set.

An essential aspect of the training process lies in the characteristics of the data that is being used. NPs are trained on multiple datasets—that is, sets of data points that have been sampled from functions f_i—to allow modelling the variability of the stochastic process. The generation (label prediction) and inference (global latent posterior update) steps are illustrated in Figure 2.7.

Finally, I highlight the differences in NP processing with respect to typical machine learning setups. An NP is trained over multiple datasets, or sets of samples $\mathcal{S}_i$ from functions $f_i \sim \mathcal{D}$, with a given training episode consisting of samples from a single such function. Sampling over the distribution of functions provides information about the variability of the stochastic process being modelled to the NP. The context set $\mathcal{C}_i$ described above is a (labelled) subset of $\mathcal{S}_i$, while the target set $\mathcal{T}_i$ is an (unlabelled) superset of $\mathcal{C}_i$, with $\mathcal{C}_i \subseteq \mathcal{T}_i \subseteq \mathcal{S}_i$. Section 5.4 further expands on each of the main components of the framework—encoder, aggregator, decoder—that are common across NP models, while presenting the one which I developed together with my collaborators.

2.2 Machine learning tasks

2.2.1 Node property prediction

In a typical supervised classification setup, a machine learning model is trained on a dataset of (*sample*, *ground-truth label*) tuples, $\mathcal{D}_{\text{train}} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_m, \boldsymbol{y}_m)\}$, and evalu-

ated on a dataset $\mathcal{D}_{\text{test}} = \{(\boldsymbol{x}_{m+1}, \boldsymbol{y}_{m+1}), \ldots, (\boldsymbol{x}_{m+n}, \boldsymbol{y}_{m+n})\}$[5] from the same distribution. During training, the labels $\boldsymbol{y}_i$ provide supervision signal; at test time, they are only used for evaluating classification performance via accuracy, F1-score or other metrics.

Besides individual features, nothing else is assumed about any of the samples—this might limit model performance in scenarios where certain relationships exist between data points. Section 2.1.7 has already introduced neural network approaches that incorporate structure during sample processing. Here, I continue the discussion by illustrating how GNN models can be used for individual-sample classification, while taking into account the intra-dataset structure.

Assume we are given a single graph $G = (V, E)$, with corresponding node features $\boldsymbol{X} \in \mathbb{R}^{|V| \times d}$ and adjacency matrix $\boldsymbol{A}$, which is derived from the collection of edges E as $A_{ij} = 1 \iff (i, j) \in E$ and $A_{ij} = 0 \iff (i, j) \notin E$. The labels available at train, validation and test time always belong to the same (disjoint) sets of nodes, regardless of the experimental setup. However, the nodes (and corresponding edges) available to the model can differ:

- in the *transductive* setup, the structure is preserved across phases, so all node features and edges are available;
- in the *inductive* setup, the model learns only from the features and edges belonging to nodes which are labelled; at test time, the structure of the graph is changed, with new nodes and corresponding edges being added.

In both cases, applying T steps of message passing to the input features yields node embeddings $\boldsymbol{H}^T$, as per Equation 2.21. These features can now be viewed as belonging to individual samples, since the relational structure in the dataset has already been exploited within the MP steps. An MLP f can be applied for downstream classification, with the predicted label of the i-th node being $\hat{\boldsymbol{y}}_i = f(\boldsymbol{H}_i^T)$.

This is the most general approach to node classification and naturally assumes no specific knowledge about the mechanism inside the MP layer. It can be noticed, however, that simply aggregating neighbourhood information might not be enough in some node classification scenarios. This observation is especially relevant in cases where the neighbourhoods are highly heterogeneous with respect to the node labels. Additionally, a single MP step corresponds to extending the aggregated neighbourhood depth by 1, so performing too many steps can yield 'washed-out' node embeddings that contain too little information about the central node.

[5] A third, validation set $\mathcal{D}_{\text{val}}$ is also often used for hyperparameter selection.

Various architectures have been developed to tackle these problems and establish state-of-the-art results on existing node property prediction benchmarks—the ones typically present in the literature are Cora, CiteSeer and PubMed [198]. More recently, the Open Graph Benchmark [85] and Wiki-CS [124] have been proposed, in order to address some of their limitations. However, the node classification scenarios present in this thesis go beyond simply learning from the same graph. Instead, those tasks involve modelling the underlying distribution of graph-generating functions—in this case, the model is trained on multiple datasets, where each one corresponds to a different graph from the distribution. This class of tasks is more challenging, as it requires representing the variability of the generative process. Together with my collaborators, I designed an NP-based architecture to tackle such tasks, which I describe and discuss in Chapter 5.

2.2.2 Graph property prediction

The previous section discussed node prediction scenarios, where the relational structure between samples is exploited by a GNN-based model. This process helps enrich the individual node representations, which are then classified independently. However, some setups require predicting a quantity for the entire graph, instead of individual nodes. For example, molecules corresponding to various chemical compounds might fall into different toxicity levels and scene graphs can originate from images of indoor or outdoor settings.

Discriminating between such categories often involves reasoning about the interactions between nodes, potentially via incorporating different types of relations dictated by the edges in the message passing scheme. This leads us to the idea of summarisation—after the nodes have been processed by MP steps and other neural transformations, we would like to aggregate all node embeddings to obtain a description of the entire input graph that is meaningful for the downstream prediction task.

One simple approach to this is taking the average across all embeddings, which results in a fixed-size representation; this can be then processed by a typical classifier such as an MLP. That is, for a given input sample $G_i = (V, E)$ with corresponding features $\boldsymbol{X}_i$ and adjacency matrix $\boldsymbol{A}_i$, T MP steps and a downstream MLP classifier f:

$$\begin{aligned} \boldsymbol{H}^T &= \mathrm{MP}^T(\boldsymbol{X}_i, \boldsymbol{A}_i) \\ \boldsymbol{r}_i &= \frac{1}{|V|} \sum_{j=1}^{|V|} \boldsymbol{H}_j^T \\ \hat{\boldsymbol{y}}_i &= f(\boldsymbol{r}_i), \end{aligned} \tag{2.24}$$

where MP^T is shorthand for $\text{MP} \circ \cdots \circ \text{MP}$ (T times). The summarisation step takes place in Equation 2.24, on the second line. One can quickly notice that useful information may be lost when aggregating all nodes into a single d-dimensional vector $\boldsymbol{r}_i$—two sets of embeddings might yield a similar summary in the embedding space, but the underlying graphs can have nodes with entirely different features!

An alternative would be to use a sum-aggregator instead of the average one. However, a fundamental issue arises in this case, when considering tasks where the number of nodes across all graphs follows a multimodal distribution or one that has a large kurtosis. Essentially, the magnitudes of the resulting embeddings will be correlated with the number of nodes in each input graph, making the prediction task more challenging (it is unlikely that the node count is informative towards solving most of the interesting tasks, but the downstream classifier would work with features that encode this quantity). Additionally, the training process would become harder, as gradients of potentially-widely varying magnitudes would be propagated through the network.

Due to these reasons, 'flat' summarisation has been improved upon lately by various approaches which tackle graph classification as the downstream task. Many such methods rely on graph pooling as a step within the GNN processing, motivated by the need to progressively compress the graph representation in a meaningful way. The remainder of this section focuses on describing the key ideas behind graph pooling, also present in the literature as graph coarsening.

Similarly to the convolution operation, pooling was initially present in CNNs. Its purpose was to iteratively reduce the dimensionality of visual features—a generalised operation has thus been described for graph-structured inputs, where the idea is to preserve the most salient features for the downstream task, while reducing the size of the graph before it gets passed on to the next layer. So far, two main approaches can be distinguished in the literature—they are illustrated in Figure 2.8.

2.2.2.1 Top-k pooling

This type of pooling layer produces an output graph that contains only a fraction r of the nodes in the input graph, discarding the rest of them. Top-k pooling is described in detail in Chapter 4, but it is important to note that the typical mechanism relies on producing a score s_i for each node via a learnable function f—the highest-scoring $r\%$ nodes are thereby preserved:

$$s_i = f(\boldsymbol{H}_i), \forall i = 1..n. \tag{2.25}$$

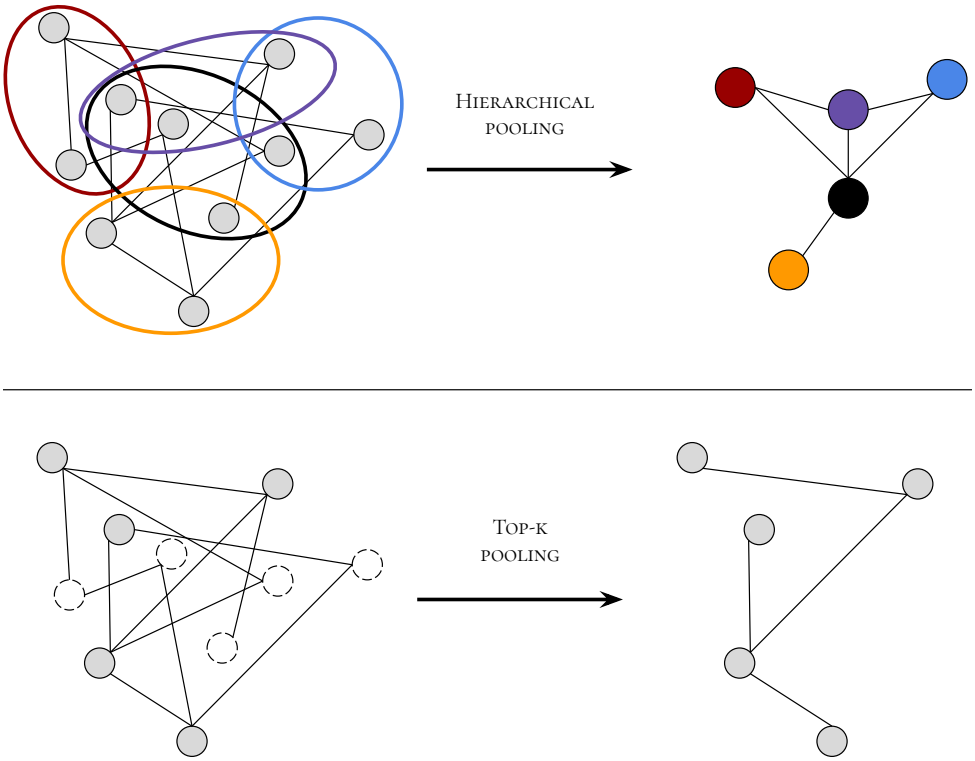

Figure 2.8: Two pooling paradigms: hierarchical clustering (*top*) and Top-k (*bottom*). In the former case, nodes in the output graph correspond to the clusters, with links between them representing cluster overlaps. In the latter, the highest-scoring $r\%$ nodes (grey) are simply passed on to the next layer, along with the links that already exist between them, while the rest of the nodes (dashed) are dropped. In the figure, $r = 0.5$.

The reasoning behind Top-k pooling is that some nodes may be more salient than others for the downstream graph prediction task. A straightforward example would be distinguishing between point-cloud graphs of *chairs* and *cars*—the pooling operation can discard numerous nodes along the object surfaces and keep the ones closer to joints and other places where different object parts combine. The resulting graph would resemble a skeleton of the initial object, which still encompasses all the information required to discriminate between classes.

Analysing Equation 2.25, we note a potential weakness to this approach. Namely, gradients will only flow through the highest-scoring node features, so there is comparatively little information being learned about the less important nodes.

2.2.2.2 Hierarchical pooling

Instead of dropping a fraction of nodes, each pooling layer now aggregates sets of nodes into clusters. The cluster assignments can be either *soft* (probabilistic assignment of a node across clusters) or *hard* (a node can only be part of a single cluster).

If the input graph has n nodes with features $\boldsymbol{H} \in \mathbb{R}^{n \times d}$ and adjacency matrix $\boldsymbol{A}$, then a cluster assignment matrix $\boldsymbol{S} \in \mathbb{R}^{n \times n_{\text{pool}}}$ will yield the following updates:

$$\begin{aligned} \boldsymbol{H}' &= \boldsymbol{S}^\top \boldsymbol{H} \\ \boldsymbol{A}' &= \boldsymbol{S}^\top \boldsymbol{A} \boldsymbol{S}. \end{aligned} \tag{2.26}$$

As in a typical CNN, graph pooling operations can be interleaved with graph convolutional (or any other variant of MP) layers. The input graph to the following layer will thus have n_{pool} nodes and be given as $(\boldsymbol{X}', \boldsymbol{A}')$. It is important to note that if $\boldsymbol{S}$ is a soft cluster assignment, then $\sum_{j=1}^{n_{\text{pool}}} S_{ij} = 1, \forall i \in 1..n$. The same statement holds for a hard cluster assignment matrix, but in a trivial manner, since $S_{ij} = 1$ for a single value of j—the index of the cluster (or node in the pooled graph) that it has been assigned to.

2.2.2.3 Summary

Both pooling paradigms reviewed in the previous sections are leveraged by contributions in this thesis. Described in Chapter 4, these works focus on graph prediction tasks and introduce novel GNN architectures which rely on progressive coarsening of the input graphs. The first one matched state-of-the-art results on graph classification benchmarks, while reducing memory requirements from quadratic to linear (in terms of the number of nodes). The second work introduces a pooling layer that produces a topologically-grounded graph summary of the input graph, while being competitive with other state-of-the-art GNN models.

2.2.3 Visual reasoning and question answering

Image classification has been extensively studied by the machine learning and computer vision communities, with LeNet [108, 109] and AlexNet [105] being milestone developments in the field. The former introduced the convolutional neural network paradigm and the second one applied it at scale, 24 years later, via modern-day GPU compute. However, in order for an AI system to gain a proper understanding of its environment (and eventually act in it), its internal model should have capabilities that go far beyond discriminating between object classes. Rather, it is necessary for subsequent processing steps that combine and reason about the outputs of the object detection phase and the relationships between them. Crucially, in a real-world environment, the 'view' of such a system will almost always contain more than one object in sight—this already yields a more complex task, when compared to the initial image classification one.

These arguments have motivated the emergence of visual reasoning—a challenging and inter-disciplinary problem space. This subfield encompasses tasks that aim to improve

Figure 2.9: Two possible images for the *'Is the woman mad?'* question, extracted from the official VQA v2 challenge[7], with correct answers *yes* and *no*, respectively. In each case, answering the question requires commonsense reasoning over the visual input—interpreting the facial expression and posture in the first case, while associating other concepts present in the image and the facial expression of the person in the second image with an overall positive ambience.

the understanding of ML systems about the world. The ultimate goal is for trained agents to be able to apply this knowledge during their interactions with a certain environment, in real-life situations. Visual reasoning systems typically receive images or videos as main input. Auxiliary modalities may also be used when solving a particular task, or they can simply contribute more information about the problem—a prime example of such a modality is natural language. Often used as a conditioning signal for the visual processing, the language input can point the system to the parts of the main input that are helpful for the downstream task. Nevertheless, some tasks require the system to have other kinds of previously-derived knowledge, such as commonsense principles—Figure 2.9 shows one example of a *yes*/*no* question that the system will be able to answer only if it carries out more involved reasoning over the image.

A widely-studied task in the visual reasoning space is Visual Question Answering (VQA). Here, the system receives an image and a question—the conditioning signal in this case—and is tasked with producing an answer, using information that is present in the visual input. Solving VQA tasks corresponds almost perfectly with some kinds of reasoning that an agent might need to perform, when encountering real-world settings which require decisions and reactions. For example, consider a self-driving car waiting for a traffic light to turn green at a big junction. A question such as *'Can you turn left?'* can only be answered in several reasoning steps: first identifying the traffic lights present in the view, then selecting the one which indicates left-turn permission (an entire challenge by itself, since arrows are often adjacent to the light on a light background, or outlined

[7] `https://visualqa.org/vqa_v2_teaser.html`

inside the traffic light on a dark background), and finally deciding whether it shows a red or a green light. If the light is yellow instead, a subsequent series of questions and reasoning steps would follow, most likely including *'Was the light previously green?'* or *'Do you have time to cross the junction before the light turns red?'*.

The main bottleneck in ML systems which tackle VQA tasks is relating concepts in language to the ones encountered in the visual input. Additional challenges are further posed by the datasets used to evaluate VQA model performance, such as inherent class imbalances present in real-world distributions. These can bias the model towards simply memorising the most frequent answer to a question which contains specific combinations, instead of aligning and meaningfully reasoning over the visual and linguistic concepts. For example, a human is much more likely to stand on a surfboard than a dog; however, the latter is still physically possible and such instances exist in datasets as rarely-occurring combinations. Therefore, the system should do more than simply exploit the bias in the data distribution, which results in bypassing the visual input and risks providing a 'blind' answer to the question.

A complementary challenge faced by the community is that of language grounding. The overarching perspective is that learning from language should not be a closed-system effort—namely, using a single modality to train systems and deriving meaning purely from the relationships between words. Rather, the aim should be to ground natural language concepts in external stimuli, such as vision and interaction with objects and the environment. Similarly, when training a visual reasoning system, it is important for the concepts to be grounded in realistic representations of the environment, but at the same time overfitting on the most likely situations should be avoided.

Several architectures have been developed to approach these challenges through the lens of VQA tasks; they typically contain sub-modules which enable the interaction of the visual and linguistic inputs within the system. Among these, two influential paradigms have been proposed: feature-wise linear modulation (FiLM) [145] and compositional attention networks leveraging memory-attention-composition (MAC) [88] cells. Chapter 3 describes VideoNavQA, a novel environment understanding task that I proposed, along with generalised FiLM and MAC models that operate across the temporal dimension to solve VQA-style tasks. Here, the visual input is no longer an image, but a video of an agent navigating within a house. The ML system encounters an additional challenge, besides the usual ones posed by VQA—it is now required to isolate the relevant concepts from across all timesteps and aggregate this information, before combining it in a meaningful way to answer the given question.

Chapter 3

Multimodal learning for environment understanding

3.1 Introduction and contribution overview

The main purpose of learning world representations is to obtain models of the environments that we live in. These models can subsequently be leveraged by agents in various downstream tasks. In real-world settings, such as houses, office spaces and cities, the information is abundant. At each point in time, certain events can take place, yielding significant changes in the environment relative to the previous timestep. Moreover, the stimuli perceived by humans—hence information streams that agents have access to—are often multimodal: we can view our surroundings (visual), but also hear (auditory) and feel (tactile) during our exploration and interaction within an environment.

Section 2.2.3 introduced visual question answering as an essential, yet challenging task for ML systems operating in the visual reasoning domain. The latter constitutes a significant part of environment understanding—humans constantly reason about their surroundings in order to make decisions and perform actions. While growing up, we learn to ground linguistic terms in the visual concepts that we encounter and get accustomed to. We can thus view the language component as a useful conditioning signal within ML systems that process rich visual streams from life-like settings.

An end-goal within this problem space is building agents that are capable of interacting with the environment and making decisions, based on their reasoning about surroundings. However, we first need to measure how complex their understanding of the world is—given the above argument on language conditioning, one way of doing this is evaluating the performance of a system on question answering (QA) tasks.

Numerous benchmarks have been proposed so far, ranging from simple question answering on a single image [7, 87, 121], where all the necessary information can be found in the visual input[1], to completing tasks which require navigation and interaction with the given environment [44, 163]. There is also a growing research sub-community focusing on this area, which has been investigating future directions and organising inter-disciplinary discussions at various workshop venues [2, 27, 185].

This chapter presents my work on environment understanding, which was started during my research internship at Mila in 2018, in collaboration with Eugene Belilovsky and Aaron Courville, who supervised me during the internship. Model design, implementation, benchmark engineering and curation were carried out by me; I also conducted all the experiments, with the exception of the Temporal Compositional Attention Networks model evaluation, which Eugene ran. Aaron proposed the initial research direction—namely, the need to isolate the navigation requirement from the visual reasoning aspect in EmbodiedQA-like tasks—and provided constant feedback and discussions throughout the project.

Our work was published as a conference paper at the 2019 British Machine Vision Conference, under the title *'VideoNavQA: Bridging the Gap between Visual and Embodied Question Answering'* [26]. A shorter, 4-page version also received commending reviews and was presented as a spotlight talk during the Thirty-third Conference on Neural Information Processing Systems in 2019, at the Visually-Grounded Interaction and Language Workshop—one of the most influential events in the language grounding and human cognition fields.

3.2 The need for a different approach to embodied question answering

The Embodied Question Answering (EQA) task proposed by Das et al. [38] concerns environment understanding through the prism of question answering. In order to answer a given question, an agent starts from a random location in a rich 3D setting and must act based solely on its egocentric input. The ultimate aim is that the agent learns to combine several capabilities—such as scene understanding, navigation and language understanding—and perform complex reasoning in the visual world. However, the first models proposed to tackle EQA combined standard vision and language methods with imitation and reinforcement learning algorithms. Initial results suggested that these techniques may struggle to achieve good performance on EQA, given its complexity

[1]There are certain VQA tasks that require external knowledge to produce the correct answer, like the recently-proposed Visual Commonsense Reasoning [205] benchmark.

Figure 3.1: High-level description of the VideoNavQA task and our approach. Each example received by the ML system contains a video of a trajectory inside a house and a question about the visual stream. The system processes this multimodal input to produce an answer. We tackle the task via already-established VQA paradigms—where the language signal is used to condition the visual processing—and build extensions of these methods that account for the temporal dimension of the visual input.

and the challenges that it poses. In order to investigate the feasibility of EQA-type tasks, I have built the VideoNavQA benchmark, which contains pairs of questions and videos procedurally generated in the House3D virtual environment. This dataset allows one to assess question-answering performance from nearly-ideal embodied navigation paths, while considering a much larger variety of questions than existing instantiations of the EQA task. I investigated the performance of several models—extensions of popular VQA methods—on the new benchmark. The results outline an initial understanding of how much VQA-style methods can help within the alternative EQA paradigm.

The most relevant tasks in this space at the time of the study were Embodied Question Answering [38] (EQA) and Interactive Question Answering [65] (IQA). Their introduction invited the research community to study the capabilities of agents in rich, realistic environments, as both tasks require navigation and reasoning to achieve success. Each of these skills typically requires a particular type of ML approach [83, 88, 128], which should nevertheless be smoothly integrated with the rest of the system that is leveraged by the embodied agent. These abilities are assessed via placing the agent at a random location in a house environment and asking it a question. Successful completion of the task thus requires the agent to knowledgeably explore the environment and reason about the visual stimuli it receives.

Accordingly, initial attempts to solve the EQA challenge [38, 39] have combined common strategies in vision (convolutional neural networks for object detection) and language (question encoding or program generation) tasks with imitation learning and reinforcement learning. However, the resulting frameworks either suffer from potentially weaker performance than when leveraging language-only models [3]—which essentially means that the visual signal is not used at all, or that it negatively interferes with the other modules—or are preceded by hand-crafted steps (such as manually defined sub-goals

and imitation learning on pre-computed expert trajectories). Indeed, even for straightforward questions referring to a single object, trained agents are often unable to advance meaningfully towards the target [39], thus producing visual streams that cannot be used to answer the query. This finding suggests that EQA is a highly challenging task which might not be best approached from this angle—further investigation is required to appropriately assess the gap between state-of-the-art and human-level performance on the EQA benchmark.

Single-image Visual Question Answering has only recently started to tackle complex reasoning questions, even in limited and controlled settings [88, 91]—it is therefore unclear whether existing methods can handle the rich video streams produced in the VideoNavQA task. On the other hand, EQA requires the system to perform navigation, which can often lead to video inputs that are uninformative with respect to the given question. A natural concern arises: can these tasks be solved with current methods, if we assume that the agent receives correct visual streams (namely, ones that contain all the information necessary to provide an answer)?

Along with my collaborators, I attempted to answer this question by designing the VideoNavQA benchmark. Illustrated in Figure 3.1, this task decouples visual reasoning from the navigation aspect in EQA. While removing the navigation and action selection requirements from EQA, I also increased the difficulty of the visual reasoning component by creating a much larger question space, which allows tackling the sort of complex reasoning questions that make QA tasks challenging [91]. I designed and evaluated several VQA-style models on the dataset, thus establishing a new way of evaluating EQA feasibility given existing methods in the visual reasoning space. At the same time, the evaluation highlighted the difficulty of the problem even in an ideal setting.

3.3 Previous related work

To the best of our knowledge, there were no previous works that positioned themselves at the intersection of the VQA and EQA paradigms. Instead, these types of tasks had been tackled from separate angles, with multiple interesting advancements in each case.

Das et al. [38] proposed the **EQA**-v1 dataset that contained only 4 types of questions (`location`, `colour`, `colour_room`, `preposition`)—these questions would always refer to a single object, which represents the navigation goal of the agent. They trained a model via imitation and reinforcement learning, revealing as a by-product that RL fine-tuning often resulted in overshooting the goal. A subsequent improvement was achieved by hierarchical policy learning with neural modular control [39]—however, this approach made use of hand-crafted sub-policies. Anand et al. [3] study the performance of

language-only baselines on EQA-v1, which result in better performance when the agent is spawned more than 10 steps away. The authors concluded that initial EQA methods struggled to exploit—and were often even impeded by—the environment.

Multi-target EQA [203] is an extension of the initial EQA task that requires reaching multiple sub-goals to answer the question (for example, the agent may be asked to compare the sizes of two objects in different rooms). The authors designed 6 types of questions and tackled the task by first decomposing each question into smaller sub-goals. The latter were deemed more easily achievable by models similar to the ones used in previous EQA works. Another EQA variant is based on photorealistic environments [186] and uses point clouds instead of RGB input. The authors built the dataset by 'porting' three of the EQA-v1 questions to the Matterport3D virtual environment.

Although the focus of this work was EQA and the House3D environment, I also note that the IQA task [65] in the AI2Thor [101] environment was introduced with similar goals. However, IQA is defined for single-room settings, which diminishes the negative effect of poor navigation.

Originally proposed by Malinowski and Fritz [121], **visual question answering** has been extensively studied during recent years, with many benchmarks being released, numerous algorithms proposed and more general studies carried out [10, 122]. The VQA dataset [7] is a prime such example—it requires natural language answers to free-form and open-ended questions about real-life images and abstract scenes. The creators of the CLEVR task [91] used a functional, program-based question representation to generate a vast range of questions from synthetic scene graphs—the latter contain between 3 and 10 objects of restricted variability (3 types, 2 sizes, 2 materials, 8 colours). Subsequently, the GQA dataset [87] was proposed to address some of the issues encountered and studied in existing datasets, including biases in the answer distribution.

Widely-adopted VQA methods include *stacked attention networks* [199], which let the question embedding to act as a query vector when attending over the visual input, *multimodal compact bilinear pooling* [54], which fuses the text and visual embeddings by multiplying them in Fourier space, *feature-wise linear modulation* [145], which emphasises the visual feature maps which are more informative for the task using question-based conditioning, its *multi-hop* extension [170], which similarly conditions feature maps by iteratively attending over the language input, and *compositional attention networks* [88], which are designed to reason about the visual input in a multi-step fashion, using a sequence of memory-attention-composition (MAC) cells.

Researchers have also started using graph neural network-based approaches for VQA tasks [136, 141, 174]. These methods operate on the relational representation of objects

in the image, instead of directly processing the raw visual data. Finally, the model that most closely resembles the ones I designed was explored in a multi-turn QA setting [138]. Here, the system is provided with a dialogue (a set of question-answer pairs) and a video. The question encoding is then used for both per-frame conditioning and computing attention over the hidden states of an LSTM, which encodes all the video frames.

Other datasets have been designed that also consider **video question answering** in settings such as films [173, 135, 112]. They are often accompanied by rich per-frame annotations, such as subtitles. However, these types of tasks are distant from the aims of VideoNavQA, instead focusing on identifying actions or other dynamic behaviors. Our task considers indoor navigation trajectories that exhibit rich visual data at each time step—this poses an additional challenge for video-question-answering systems. The latter are now required to isolate the relevant information from a large set of visual concepts present in all frames and perform more complex reasoning to answer the extensive variety of questions designed.

3.4 A new benchmark for embodied reasoning

I have constructed a new benchmark to study the capabilities of VQA-based methods within a novel variant of the EQA task. Here, the agent is required to answer a question while having access to a near-optimal trajectory—that is, the given trajectory corresponds to a natural one with sufficient information in the video signal to answer the question. The VideoNavQA task can be deemed complementary to the Habitat Challenge [159], where the focus is on the navigation aspect, instead of question answering.

I used the House3D virtual environment [190] to generate approximately 101,000 pairs of videos and questions. The dataset contains 28 types of questions that belong to 8 categories (see Figure 3.2), with 70 possible answers in total. Each question type is associated with a template that facilitates programmatic generation using ground-truth information extracted from the video.

The complexity of VideoNavQA questions is far beyond that of other similar tasks which use this generation method (such as CLEVR). Our questions involve single or multiple object/room existence, object/room counting, object colour recognition and localisation, spatial reasoning, object/room size comparison and equality of object attributes (colour, room location). The full list of question types and counts can be found in Appendix A.1.

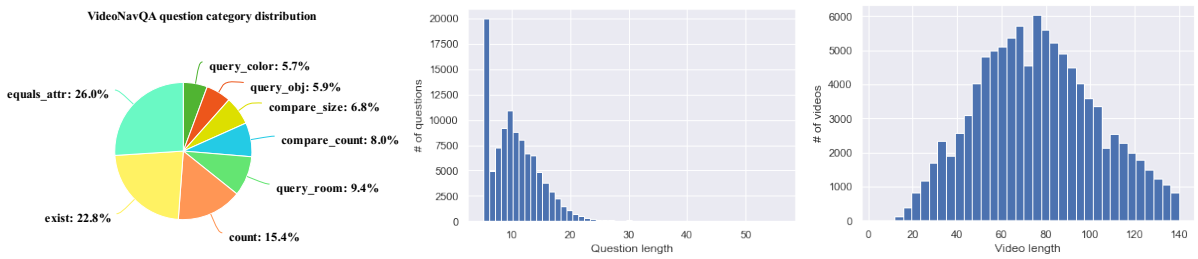

Figure 3.2: Insights into the VideoNavQA dataset distribution. (*Left:*) Proportions for each of the eight question categories. Equality and existence questions account for nearly half of the dataset. (*Middle:*) Question length distribution. The maximum length is 56 and approximately a fifth of the questions contain five words; however, the distribution is bimodal and considerably long-tailed, beyond 20 words. (*Right:*) Video length distribution. The maximum length is 140, with a normal distribution centered around 70 frames, which is most likely an effect of the random sampling of start and end locations for each trajectory.

	Houses	Samples
Train	622	84990
Validation	65	8755
Test	56	7587

Table 3.1: Dataset split statistics. The VideoNavQA benchmark contains approximately 100000 samples. The three sets of house environments are disjoint and correspond to the splits used in the EQA-v1 dataset. This ensures that generalisation performance is assessed in entirely novel visual settings (that is, trajectories inside previously unseen houses) and that the visual variety is considerable across the dataset.

3.4.1 Visual information

Environments and scene representation The House3D environment is based on indoor scenes from the SUNCG dataset [168]. Table 3.1 shows the number of houses present in each of the three dataset splits and the corresponding number of examples (question-video pairs). A house does not appear in more than a single split and 150 videos, to ensure large and consistent variability in the visual information.

Video generation and ground-truth extraction VideoNavQA examples are constructed by first generating the video component. I used the grid-based representation available in the House3D environment to compute the shortest path between arbitrary locations in two different rooms of the house. To obtain the video, I rendered the shortest-path trajectory inside House3D. The output corresponds to what an agent would see in an EQA setting when exploring the house—with the added benefit that this trajectory is already sensible from a navigation perspective.

EQA-v1 (Q types: 4)	What room is the <OBJ> located in? What colour is the <OBJ> in the <ROOM>?
VideoNavQA (Q types: 28)	Are both <attr1> <OBJ1> and <attr2> <OBJ2> <colour>? How many <attr> <OBJ> are in the <ROOM>? Is there <art> <attr> <OBJ>?

Table 3.2: A sample comparison between the question templates found in EQA-v1 and the more complex existence, counting and comparison templates found in VideoNavQA. EQA-v1 questions only enquire about a single object, whereas VideoNavQA questions may require reasoning about a pair of objects or identifying more objects of a certain kind across a trajectory.

The ground-truth information is then obtained by parsing each video frame and stored for subsequent question generation purposes. I used the SUNCG semantic rendering mode in House3D to identify the objects visible in each frame. By linking them via depth rendering to the current room the agent is in, or to an adjacent one, I collated all objects and rooms that are seen on the trajectory and, consequently, in the video.

3.4.2 Questions

Functional form representation In a manner which is similar to other synthetic benchmark datasets (EQA-v1 [38], CLEVR [91]), I generated questions starting from functional, template-style representations (for example, *'How many <attr> <obj_type-pl> are in the <room_type>?'*). Once the video for the corresponding trajectory has been generated and analysed, the tags can be instantiated with ground-truth information from the video. Moreover, the correct answer can be easily determined, by executing the corresponding functional program on the ground-truth collection of objects and rooms that have been seen or visited on the trajectory. This involves performing a sequence of basic operations (such as `filter()`, `count()`, `get_attr()`) on the ground-truth collection. Table 3.2 shows examples of question templates from VideoNavQA, while contrasting them with the more restricted ones encountered in EQA-v1.

Generation This process starts with randomly choosing one of 28 question templates to be instantiated. A valid question requires tags to be instantiated with ground-truth values. To illustrate this idea, if the template contains a *<room_type>* tag and the ground-truth set obtained from the video trajectory only contains a kitchen and a living room, then the only valid instantiations are {*kitchen*, *living room*}. With this in mind, I built sets of possible values for each tag in the question template. To generate a valid (*question*, *answer*) pair, I randomly assign each tag a value from its set, then ran the template functional program to compute whether the question is valid and can be answered using

the ground-truth. I illustrate the process using the template *'What colour is the <attr> <obj_type>?'*, with the associated program:

`input(objs)→filter(obj_type)→filter(attr)→unique()→get_attr(colour)`

I first select the objects seen on the trajectory from the ground-truth collection. Next, I filter by the instantiated object type, then by the instantiated attribute (enforced not to be a colour during the tag value assignment). Finally, I ensure that the result is unique (that is, the question is unambiguous) and retrieve the colour of the object.

3.5 Methods

The VideoNavQA dataset was designed to tackle the EQA task from a different perspective, which requires a smaller degree of fusion among different classes of methods. I now describe the architectures I designed and used to establish more realistic expectations on EQA performance, thereby obtaining initial results on VideoNavQA. These models include several essential baselines and new ones inspired by previous successes in visual question answering and computer vision.

3.5.1 Single-modality

3.5.1.1 Language

Question-only models have proved remarkably effective in EQA, often performing better than the complex initial approaches [3]. I therefore included two simple yet powerful language-modelling methods in the evaluation: a *1-layer LSTM* [81] and an *bag-of-words* (*BoW*) [154] model. Both models use an initial embedding layer, which learns a separate representation for each of the 134 vocabulary words. The BoW averages all question word encodings and forwards the result to a linear layer, while the LSTM encodes the resulting sequence into a vector. Both baselines reveal the inherent biases that exist in the environment concept distribution, thereby placing a lower bound on the desired performance of models that can usefully exploit visual information.

3.5.1.2 Vision

Learning to answer a question solely from visual inputs is not expected to perform better than providing the most frequently-occurring answer (in VideoNavQA, approximately 66% of the questions require a *Yes*/*No* answer). To confirm this behaviour, I evaluated the performance of two popular video-processing models:

1. V-CNN2D: a per-frame processing VGG-style [165] convolutional neural network

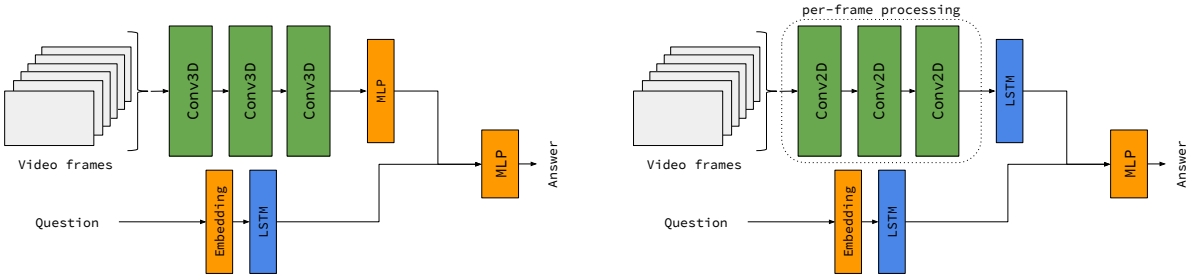

Figure 3.3: (*Left:*) Concat-CNN3D processes the entire video at once, extracts a visual representation and concatenates it with the question embedding. The result gets passed through the MLP classifier to produce an answer. (*Right:*) Concat-CNN2D extracts frame-wise features, then aggregates them via an LSTM—the rest of the processing is the same as for Concat-CNN3D.

with LSTM integration across time. Here, the CNN extracts frame features $\boldsymbol{x}_t$ and the LSTM is applied to the resulting sequence $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$. Finally, a linear layer takes the output of the LSTM after T steps and outputs a probability distribution over all possible answers via softmax.

2. V-CNN3D: a C3D-like [177] CNN, consisting of {3D-Convolutional, Max-Pooling, BatchNorm} blocks, followed by a 3-layer MLP classifier with batch normalisation applied after the first two layers. The convolutional and linear layer outputs are passed through ReLU activation functions.

3.5.2 Multiple-modality

3.5.2.1 Concatenation models

Based on single-modality results, I integrated the best language baseline (LSTM) with each vision model, in order to obtain joint representations of the obtained features. This was achieved by concatenating the final question representation with the respective video representations (from Concat-CNN2D/3D, shown in Figure 3.3). The result was then passed through an MLP classifier that predicts the answer via softmax.

3.5.2.2 FiLM-based per-frame reasoning

Feature-wise linear modulation (FiLM) [145] was a previous state-of-the-art method for visual question answering tasks. This paradigm uses the question embedding as a conditioning signal that scales and shifts the feature maps within a CNN pipeline.

Illustrated in Figure 3.4, this method can be viewed as a general conditioning layer, but with immediate applicability to a language conditioning input—in our case, we are conditioning the visual processing using the question that the system needs to answer.

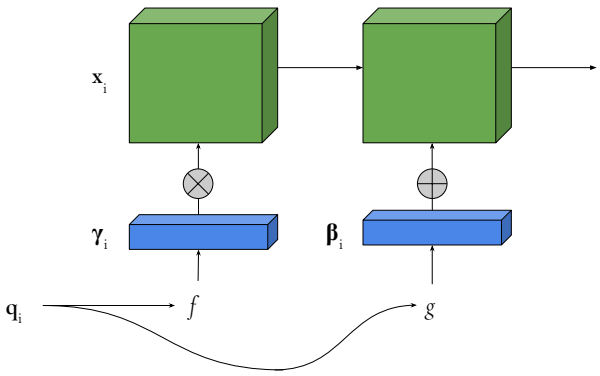

Figure 3.4: An illustration of the affine transformation computed inside the FiLM layer. The question embedding gets passed through two functions f and g, which yield scale and shift parameters $\boldsymbol{\gamma}_i$ and $\boldsymbol{\beta}_i$, respectively. The former is multiplied element-wise with the visual features $\boldsymbol{x}_i$, followed by an element-wise addition with the shift values $\boldsymbol{\beta}_i$. Intuitively, this operation ensures that the intermediate network representations get (de-)emphasised according to the conditioning signal. End-to-end training ensures that these parameters are learned as required by the downstream task.

Specifically, a FiLM layer takes as input a conditioning signal $\boldsymbol{q}_i$ and learns functions f and g which compute scale and shift values γ_{ic} and β_{ic}, respectively. These values are computed independently for each channel c of the current intermediate network features $\boldsymbol{x}_i$ that we wish to modulate:

$$\gamma_{ic} = f(\boldsymbol{q}_i)_c, \quad \beta_{ic} = g(\boldsymbol{q}_i)_c. \tag{3.1}$$

In turn, $\boldsymbol{\gamma}_i$ and $\boldsymbol{\beta}_i$ enable the FiLM layer to output an affine transformation of the network features $\boldsymbol{x}_i$:

$$\text{FiLM}(\boldsymbol{x}_i) = \boldsymbol{\gamma}_i \cdot \boldsymbol{x}_i + \boldsymbol{\beta}_i. \tag{3.2}$$

Intuitively, this modulates the neural network intermediate representations towards ones that are useful for the downstream task—in particular, in a VQA setting, the network is encouraged to focus on the parts of the image that help answer the question. In practice, f and g are implemented as neural networks.

Here, I extended FiLM to address the temporal dimension introduced by VideoNavQA. Figure 3.5 (left) shows each of the T video frames being processed independently by a fixed number of ResBlocks [145]. The conditioning mechanism ensures that the visual features relevant for question answering are propagated to the final frame representation. Both functions f and g are implemented by neural networks consisting of an LSTM followed by a linear layer. This can be viewed as an encoder-decoder architecture, where the LSTM produces a representation of the entire question and the linear layer decodes this embedding into a suitable affine transformation of the current ResBlock output.

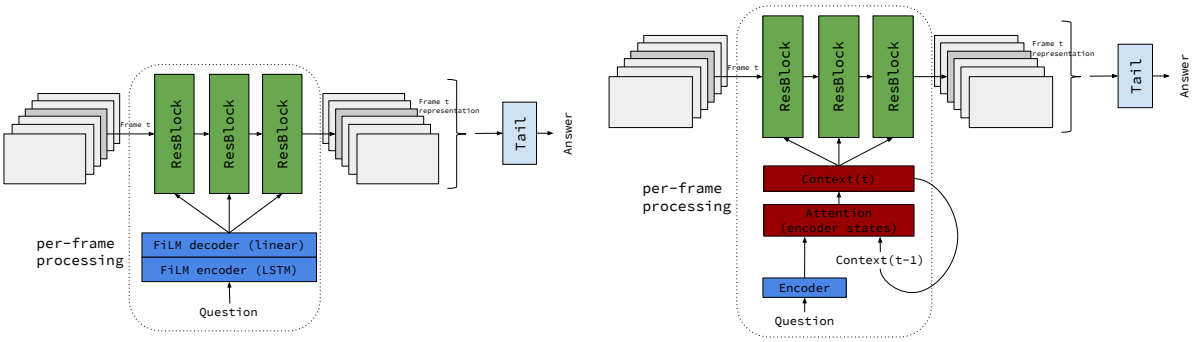

Figure 3.5: (*Left:*) Per-frame FiLM model. Each video frame is processed separately by the ResBlocks; then, all of the frame representations are aggregated and passed through the classifier, which produces an answer. The aggregator is based either on attention or global max-pooling mechanisms. (*Right:*) Temporal Multi-hop model. Each video frame is processed by the ResBlocks: the FiLM parameters are computed from the current attention context, which is initialised with the one from the previous frame and attends over the hidden states of the question encoder. Temporal summarisation is achieved via global max-pooling.

A series of visual features $\boldsymbol{x}_t$ across all frames were thereby obtained. They were then aggregated via two mechanisms corresponding to the model variants that I evaluated:

- *attention* (FiLM AT variant): I applied a linear transformation h to the features from each frame which results in $\boldsymbol{x}'_t$, then used recurrent attention [9, 34] (please refer to Section 2.1.6 for a detailed description of this method) to obtain intermediate attention contexts $\boldsymbol{c}_t$; these were concatenated (denoted by $\|$) to produce a final encoding of the video conditioned on the question ($\boldsymbol{x}_{\text{aggr}}$):

$$\begin{aligned} \boldsymbol{x}'_t &= h(\boldsymbol{x}_t), \forall t = 1..T \\ \boldsymbol{x}_{\text{aggr}} &= \|_{t=1}^{T} \boldsymbol{c}_t; \end{aligned} \tag{3.3}$$

- *global max-pooling* (FiLM GP variant): I first applied a 1×1 convolution with ReLU activations to obtain representations $\boldsymbol{x}'_t$, which were then passed through a feature-wise (h, w) max-pooling operation over all frames (t') to obtain the final encoding $\boldsymbol{x}_{\text{aggr}}$:

$$\begin{aligned} \boldsymbol{x}'_t &= \text{ReLU}(\text{Conv}_{1\times1}(\boldsymbol{x}_t)), \forall t = 1..T \\ (\boldsymbol{x}_{\text{aggr}})_{hw} &= \max_{t' \in 1..T} (\boldsymbol{x}'_{t'})_{hw}. \end{aligned} \tag{3.4}$$

Finally, the aggregator output was fed to a linear classifier that predicts the correct answer. The decision to use these two types of classifiers is motivated by the rich representation of the video across time, from which visual reasoning methods need to

select the information required to answer the question—this is often spread across only a few frames. Both attention and global max-pooling are typically effective at selecting the important features from a sequence, thus being reasonable choices in this scenario.

3.5.2.3 Temporal multi-hop FiLM

The multi-hop extension of FiLM [170] modulates feature maps at a certain level in the CNN hierarchy (namely, the output of a particular ResBlock), by attending over the hidden states of the question encoder. The attention computation is initialised with the context vector which was obtained at the previous level—this allows scaling the approach to settings with a longer input sequence, such as a dialogue.

VideoNavQA requires that the visual reasoning process includes an additional temporal dimension. I thereby designed a temporal multi-hop model, visually depicted in Figure 3.5 (right), to condition the video features using the question input. FiLM parameters at frame t are computed for all ResBlocks, by first encoding the question using an LSTM. The decoding step then initialises the attention context with the one from frame $t-1$ ($\boldsymbol{c}_{t-1}$) and performs attention over the question encoder hidden states. Finally, the output weights $\alpha_{tt'}$ yield the new context vector $\boldsymbol{c}_t$—all of this taking place according to the approach described in Section 2.1.6. As per the previous design, the final decoding operation passes the context vector through a linear layer. Therefore, the FiLM parameters at each time step depend on what has already been computed for previous frames, which models the temporal structure of the input.

3.5.2.4 Temporal Compositional Attention Networks

Compositional Attention Networks [88] have achieved excellent performance in several VQA tasks [10, 87, 88]. They use a sequence of MAC cells for processing each question-and-image pair. The computation performed by a single cell is depicted in Figure 3.6 and comprises three interacting units: control, read and write. I first describe the input processing done by our model, then the computations taking place at each step within the corresponding MAC cell (according to the original paper). Finally, I explain how our Temporal MAC model processes the input video and question to predict the answer.

Input processing—question Each word in the question from the current example is passed through a learnable embedding layer, which produces a corresponding fixed-size feature vector. The resulting sequence is then passed through a bidirectional LSTM which yields (a) context words $\boldsymbol{cw}_s$, with $s \in 1..S$, where S is the length of the question, and (b) an overall question representation $\boldsymbol{q} = [\boldsymbol{h}_1^{\text{bck}}, \boldsymbol{h}_S^{\text{fwd}}]$, where $\boldsymbol{h}_*^*$ are the final hidden states from the forward and backward passes.

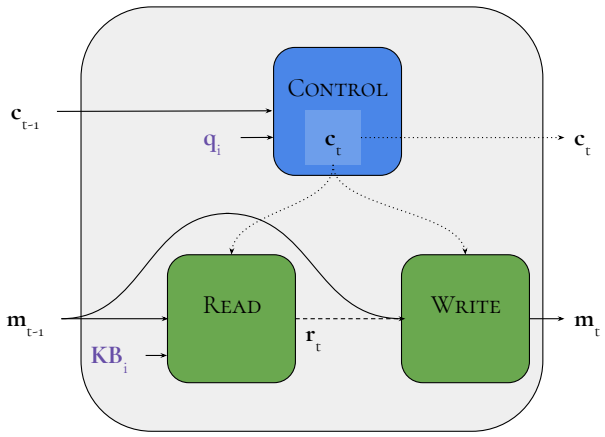

Figure 3.6: Illustration of the processing that takes place inside a MAC cell. The current question embedding $\boldsymbol{q}_i$ serves as input to the *control unit* within each cell, along with the control output from the previous cell; this allows current decisions to take into account and depend on previous ones. The visual features $\mathbf{KB}_i$ represent the knowledge base and are input to the *read unit*, which is governed by the current control signal $\boldsymbol{c}_t$. The previous memory representation $\boldsymbol{m}_{t-1}$ is used in conjunction with the knowledge base to derive useful features $\boldsymbol{r}_t$ for the current reasoning step. These are finally used by the *write unit* to update the memory and produce $\boldsymbol{m}_t$.

Input processing—video Each frame in the input video is passed through a pre-trained feature extractor. Its outputs are then fed into a convolutional neural network with 3 layers and ELU activation functions, which are defined as:

$$\text{ELU}(x) = \begin{cases} x, & x \geq 0 \\ \exp(x) - 1, & x < 0. \end{cases} \tag{3.5}$$

The final output is the knowledge base $\boldsymbol{k}$ used within MAC cells, as described below.

MAC cell—control unit This block processes the question at each step t via an attention mechanism and updates the control state $\boldsymbol{c}_{t-1}$—the result $\boldsymbol{c}_t$ encodes the reasoning operation that should be performed at the current step. The following equations summarise the control block:

$$\begin{aligned} \boldsymbol{ctl}_t &= \boldsymbol{W}_{\text{ctl}}(\boldsymbol{c}_{t-1} \parallel \boldsymbol{q}_t) + \boldsymbol{b}_{\text{ctl}} \\ a_{ts} &= \boldsymbol{W}_{\text{at}}(\boldsymbol{ctl}_t \odot \boldsymbol{cw}_s) + b_{\text{at}} \\ v_{ts} &= \text{softmax}(a_{ts}) \\ \boldsymbol{c}_t &= \sum_{s=1}^{S} v_{ts} \cdot \boldsymbol{cw}_s, \end{aligned} \tag{3.6}$$

where $\boldsymbol{W}_{\text{ctl}} \in \mathbb{R}^{d \times 2d}, \boldsymbol{b}_{\text{ctl}} \in \mathbb{R}^{d}, \boldsymbol{W}_{\text{at}} \in \mathbb{R}^{1 \times d}$, $\odot$ represents the dot-product operation and $\|$ denotes concatenation. The control block essentially attends over the question word contexts $\boldsymbol{cw}_s$ (where $s = 1..S$), using as conditioning signal the previous control state $\boldsymbol{c}_{t-1}$ and the current encoding for the question $\boldsymbol{q}_t$, then weights the contexts to produce a new control state $\boldsymbol{c}_t$.

MAC cell—read unit This block is conditioned on the output of the control block at the current step $\boldsymbol{c}_t$, which it uses to query the knowledge base $\boldsymbol{k}$ (image features) and retrieve the information relevant to the t-th processing step, $\boldsymbol{r}_t$. The equations below encompass the transformations required to produce $\boldsymbol{r}_t$:

$$\begin{aligned}
\boldsymbol{I}_{thw} &= (\boldsymbol{W}_m \boldsymbol{m}_{t-1} + \boldsymbol{b}_m) \odot (\boldsymbol{W}_k \boldsymbol{k}_{hw} + \boldsymbol{b}_k) \\
\boldsymbol{I}'_{thw} &= \boldsymbol{W}_I (\boldsymbol{I}_{thw} \parallel \boldsymbol{k}_{hw}) + \boldsymbol{b}_I \\
\boldsymbol{a}_{thw} &= \boldsymbol{W}_{\text{at}} (\boldsymbol{c}_t \odot \boldsymbol{I}'_{thw}) + \boldsymbol{b}_{\text{at}} \\
\boldsymbol{v}_{thw} &= \text{softmax}(\boldsymbol{a}_{thw}) \\
\boldsymbol{r}_t &= \sum_{h=1,w=1}^{H,W} \boldsymbol{v}_{thw} \cdot \boldsymbol{k}_{hw},
\end{aligned} \tag{3.7}$$

where $\boldsymbol{W}_m, \boldsymbol{W}_k, \boldsymbol{W}_{\text{at}} \in \mathbb{R}^{d \times d}, \boldsymbol{W}_I \in \mathbb{R}^{d \times 2d}$ and $\boldsymbol{b}_m, \boldsymbol{b}_k, \boldsymbol{b}_{\text{at}}, \boldsymbol{b}_I \in \mathbb{R}^{d}$. The first two steps allow information from the previous step ($\boldsymbol{m}_{t-1}$) and new, possibly unrelated, information ($\boldsymbol{k}_{hw}$), respectively, to be incorporated in the current reasoning step. Attention is then computed over all knowledge base elements, in order to extract the ones necessary for the reasoning operation. This results in a final, weighted representation $\boldsymbol{r}_t$.

MAC cell—write unit This block computes the result of the t-th reasoning step and uses it to update the memory state $\boldsymbol{m}_t$.

$$\begin{aligned}
\boldsymbol{m}_t^{\text{prev}} &= \boldsymbol{W}_{\text{prev}} (\boldsymbol{r}_t \parallel \boldsymbol{m}_{t-1}) + \boldsymbol{b}_{\text{prev}} \\
a_{tt'} &= \text{softmax}(\boldsymbol{W}_{\text{at}} (\boldsymbol{c}_t \odot \boldsymbol{c}_{t'}) + b_{\text{at}}) \\
\boldsymbol{m}_t^a &= \sum_{t'=1}^{t-1} a_{tt'} \cdot \boldsymbol{m}_j \\
\boldsymbol{m}_t^w &= \boldsymbol{W}_p \boldsymbol{m}_t^{\text{prev}} + \boldsymbol{W}_a \boldsymbol{m}_t^a + \boldsymbol{b}_m \\
c_t^w &= \boldsymbol{W}_w \boldsymbol{c}_t + b_w \\
\boldsymbol{m}_t &= \sigma(c_t^w) \boldsymbol{m}_{t-1} + (1 - \sigma(c_t^w)) \boldsymbol{m}_t^w,
\end{aligned} \tag{3.8}$$

where $\boldsymbol{W}_{\text{prev}} \in \mathbb{R}^{d \times 2d}, \boldsymbol{W}_{\text{at}}, \boldsymbol{W}_w \in \mathbb{R}^{1 \times d}, \boldsymbol{W}_p, \boldsymbol{W}_a \in \mathbb{R}^{d \times d}$ and $\boldsymbol{b}_{\text{prev}}, \boldsymbol{b}_m \in \mathbb{R}^{d}$. The purpose of the first operation is to integrate the current relevant information $\boldsymbol{r}_t$ with the previous memory $\boldsymbol{m}_{t-1}$. The following steps ensure that the final operation, which produces $\boldsymbol{m}_t$,

uses the result of this integration ($\boldsymbol{m}_t^{\text{prev}}$) and an attention-weighted summary of all previous memory states ($\boldsymbol{m}_t^a$). The latter is useful in cases where the reasoning required is not sequential—such as for the question *'Are there more blue cubes than red spheres?'*, as opposed to *'What colour is the sofa next to the TV?'*. The final update uses a gating operation conditioned on the current control state $\boldsymbol{c}_t$. Gating ensures that the number of steps can be dynamically adjusted, based on the complexity of the current question.

Temporal processing I have extended compositional attention networks to handle processing across the temporal domain. I started by applying a sequence of MAC cells to the per-frame visual features that were previously extracted during input processing. The final MAC cell outputs $(\boldsymbol{m}_T)_i$, where T is the number of cells and i is the index of the current frame in the video. I then used an LSTM to integrate these representations across the entire duration of the video. Finally, the LSTM output at the final step predicts the correct answer via a 2-layer MLP with ELU activations after the first layer.

3.6 Experiments

We evaluated all previously-described models on the VideoNavQA task, producing an initial expectation of the feasibility of EQA-type tasks. Accuracy was used to compare the overall performance between different architectures. Furthermore, in-depth analysis was also carried out for each of the models, which determined their respective strengths across different question categories and types. We trained all models with the Adam optimiser [95] and monitored the validation accuracy. Hyperparameters were chosen by evaluating several combinations of layer hidden sizes, number of residual blocks and classifier dimensions (detailed in Appendix A.2).

3.6.1 Models evaluated

Language-only models The recurrent model consists of an embedding layer of 512 units and an LSTM with 128 hidden units. The BoW model uses an embedding size of 128. We employed a batch size of 1024 and learning rates of $5e^{-5}$ and $1e^{-5}$, respectively.

Video-only models The V-CNN3D model has 3 {convolutional, max-pool, BatchNorm} blocks with 64, 128, and 128 output channels, respectively, kernel size $(1, 2, 2)$ for the first convolutional layer and $(4, 4, 4)$ subsequently. The classifier has 2 linear layers with 2048 and 128 hidden units, respectively. BatchNorm [89] and ReLU [62] activations were used at each layer. The V-CNN2D model is built on a VGG configuration with 5 {conv, max-pool} blocks having 16, 32, 64, 128 and 128 channels, respectively, and a kernel size of 2.

FiLM-extended models The models that process each frame in a FiLM fashion have 4 (GP) or 5 (AT) ResBlocks with 1024 channels, preceded by a 1×1 convolution on the input. The attention mechanism for FiLM AT has 128 hidden units, whereas the global max-pooling classifier obtains 32 channels via the 1×1 convolution. The temporal multi-hop model has 3 ResBlocks and 64 tail channels. It was trained with a learning rate of $5e^{-5}$ and a batch size of 16, whereas the other two used $1e^{-4}$ and 32.

Temporal MAC Each question token is embedded into a 128-D vector and the output dimension of the question encoder is 512. The model contains 6 MAC cells corresponding to at most 6 reasoning steps. The LSTM that integrates the reasoning outputs for all frames has input and hidden sizes of 1536, and the classifier that produces the answer is a 2-layer MLP with 1536 and 1024 input features, respectively, where the first layer has ELU activations. Eugene adapted the ramp-up/down Adam learning schedule often used in VQA studies [24], increasing the learning rate to $1e^{-4}$ in the first two epochs and then decaying it to $1e^{-5}$ after the 10^{th} epoch (training is done for 15 epochs).

Video dimensionality The raw videos have a $140 \times 3 \times 160 \times 208$ dimensionality, which makes learning infeasible time-wise and restricts model capacity. Instead, I extracted visual features from an object detector that was pre-trained on a set of 2000 frames not part of the dataset and initialised with the 10^{th} layer output of a Faster R-CNN [155]. The model has 3 {conv$\times$2, BatchNorm, max-pool} blocks with 512 output maps for each layer and a softmax classifier with a 1024-unit linear layer. I did not notice any significant difference in validation accuracy when training the Concat-CNN3D model with and without pre-trained features. I also adopted a randomised sub-sampling strategy when reducing the maximum video length from 140 to 35. On each iteration, I replaced every block of 4 frames with a single, randomly-picked one—which also encourages more robustness and invariance to changes between nearby frames.

3.6.2 Model performance

As expected, video-only baselines are only able to predict the most-commonly occurring answer (*Yes*/*True*). The LSTM emerges as the most powerful language model, scoring 7.5% higher than the bag-of-words. Rather surprisingly, the Concat-CNN2D vision-and-language model outperforms all other models, obtaining an accuracy of 64.47%. The overall performance of the Concat-CNN3D and FiLM AT models is roughly similar, while FiLM GP falls behind Concat-CNN2D by approximately 0.7%. In turn, temporal multi-hop scores about 0.3% lower, but still surpasses the best language baseline by 7%.

Model	All types	Yes/No	Other	Count
BoW	49.02	57.67	30.57	40.21
LSTM	56.49	68.36	35.27	38.90
V-CNN3D	33.29	-	-	-
V-CNN2D	33.62	-	-	-
Concat-CNN3D	64.00	72.99	49.12	49.10
Concat-CNN2D	64.47	73.50	49.20	49.59
FiLM-GP	63.79	72.91	47.71	50.00
FiLM-AT	64.08	72.93	49.54	49.26
Temporal multi-hop	63.53	71.81	49.54	50.16
MAC	62.32	69.02	51.37	50.99

Table 3.3: Evaluation results on the VideoNavQA test set, reported in terms of percent accuracy for language-only, video-only and multiple-modality models. We also used standard VQA reporting of Yes/No, other, and number categories [7]. This allowed a more thorough investigation of model performance and assessing the relative complexity of each type of task, considering per-category biases in the answer distributions.

Overall, these results show a significant margin over the language-only capabilities of the LSTM and BoW. We can thus deduce that the vision-and-language models initially designed and evaluated on VideoNavQA are capable of exploiting the visual information available in the environment. This finding represents an initial validation of the feasibility of the VideoNavQA task, as it ensures that the way in which the dataset was curated allows the visual reasoning process to be generalisable to unseen environments. In the future, a more focused investigation of novel VQA techniques is required, in order to properly model and account for the rich dimensionality and contextual information encountered in the videos.

3.6.3 Analysis by question category

In Figure 3.7, I provide a more detailed analysis of how each model performs on the test set for each of the eight question categories. This investigation reveals the specific capabilities developed on VideoNavQA sub-tasks. All models score highest on *existence* questions, with FiLM GP having an edge over others, whereas *identifying object types* seems to be the hardest task: Concat-CNN3D performs worst, whereas FiLM AT achieves the highest accuracy, closely followed by Multi-hop. MAC is by far the best at *identifying the room locations of objects* and obtains the strongest performance on *counting* as well. Here, all other models score roughly 50%. This is relatively expected, since MAC has been previously shown to significantly surpass FiLM on VQA tasks. When *comparing object attributes* (such as room location, colour), most models achieve around 73%, with Concat-CNN2D obtaining a 1% edge and MAC, a 5% drop. MAC scores highest once

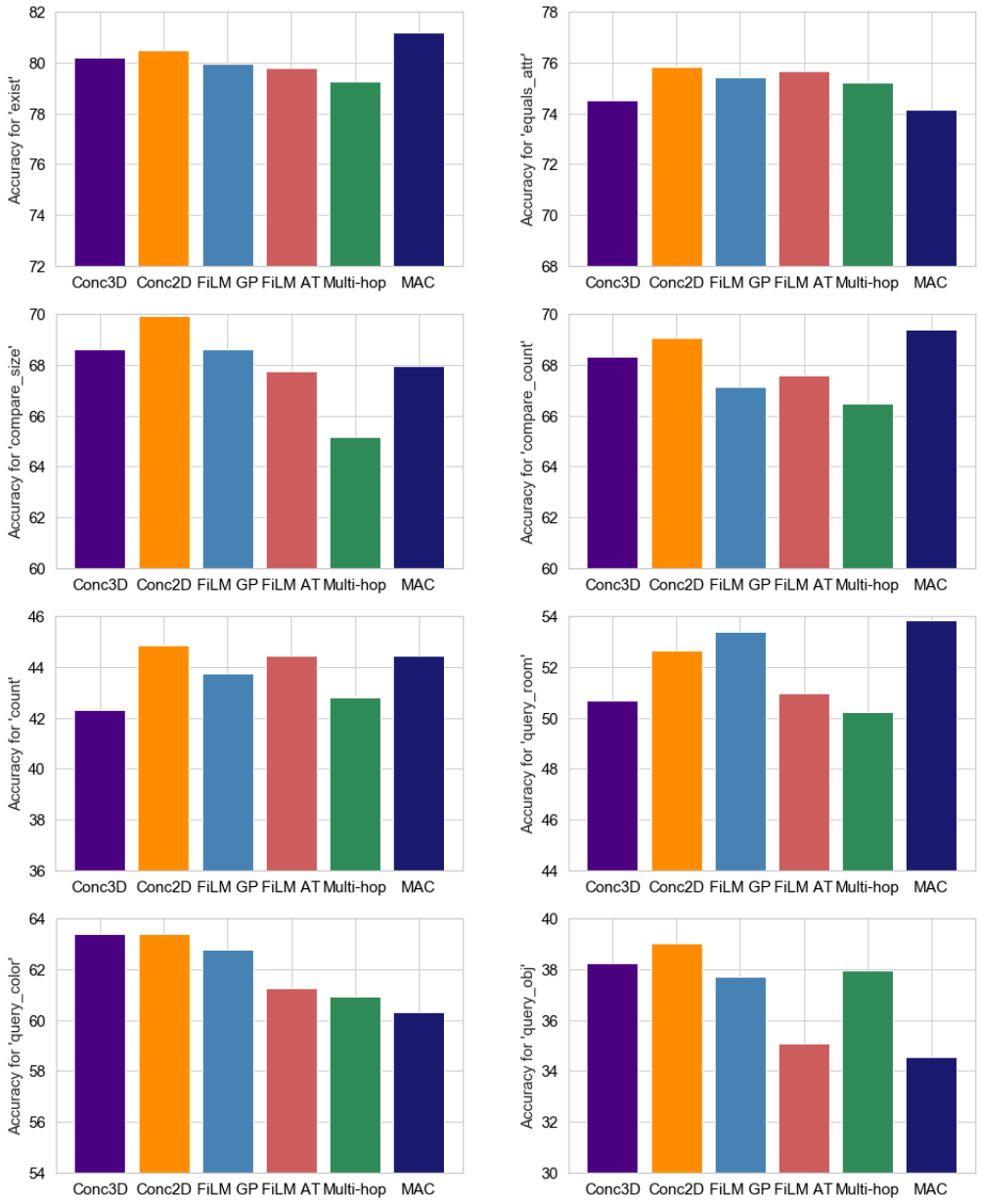

Figure 3.7: Comparative performance of all vision-and-language models on each question category (top to bottom, left to right): *exist*, *equals_attr*, *compare_size*, *compare_count*, *count*, *query_room*, *query_color*, *query_obj*. Questions from the first 4 categories have binary answers (*Yes*/*No*), whereas the latter ones are answered by integer counts, room types, colours and object types, respectively.

again when *identifying colours*, with Concat-CNNs achieving the next best score of 58% and FiLM AT performing worst, just under 56%. FiLM GP does relatively well at *comparing object/room sizes*, followed by FiLM AT, while Multi-Hop obtains a weaker result of around 71% and MAC achieves only 68%. Alternatively, FiLM GP and MAC do much worse than all the other models when *comparing counts* and Concat-CNN2D considerably outperforms them, with an overall performance gap of 5%. The motivation behind this result might stem from the fact that the question category had been relatively unexplored in VQA tasks that were visually-realistic—that is, with more variety and complexity than CLEVR-style visual inputs.

3.7 Discussion

I have introduced the VideoNavQA task as a means to achieve a clearer understanding of what is achievable in embodied QA. Here, the environment context is visually-rich and suitable methods need to be integrated to build agents that can reason, navigate and act accordingly. Our task re-imagines the EQA paradigm—we have trivialised the navigation aspect by providing the agent with nearly-optimal trajectories. Meanwhile, the difficulty of the visual reasoning requirement was significantly increased via designing 28 question types spanning 8 categories, which goes far beyond the EQA task set.

By reporting initial results from a variety of single-modality and integrative models, I have placed a lower bound on the performance achievable in the VideoNavQA task. Nevertheless, results are not widely different across architectures—this indicates an emerging need for further investigation of QA models that are better able to select the required visual information, bridging the gap to human performance. By first tackling the VideoNavQA task effectively, researchers in the vision-and-language navigation domain will encounter clearer and better-measured progress when attempting to integrate navigation into the EQA task, gradually reaching a better solution.

3.8 Additional related research

After the completion of this project, I have also been involved in other collaborations on visual reasoning tasks—namely, scene graph extraction from an image, which can be thought of as an essential step in identifying the concepts present in the view of an agent:

- *'Graph Density-Aware Losses for Novel Compositions in Scene Graph Generation'* [100]: published at the 31st British Machine Vision Conference in 2020.

- *'Generative Graph Perturbations for Scene Graph Prediction'* [99]: presented as a workshop paper at the Object-Oriented Learning Workshop, during the Thirty-eighth International Conference on Machine Learning in 2020.

Both studies aimed to make image-to-scene-graph generation systems more robust in the presence of dataset distribution biases, ameliorating their effects when predicting relationships between the objects identified in the input image.

As a third author on both works—preceded by Boris Knyazev and Harm de Vries, and followed by Graham Taylor, Aaron Courville and Eugene Belilovsky—my contributions consisted of suggesting model changes and approaches (supported by my knowledge of graph neural network architectures), implementing a data pipeline for running experiments on the GQA dataset and contributing to the paper writing process.

3.9 Summary

This chapter has presented an approach to modelling world representations via visual reasoning paradigms. I have proposed a novel benchmark, VideoNavQA, which views the existing Embodied QA task from a different perspective. VideoNavQA removes the navigation aspect that poses a significant challenge in agents and focuses on investigating the QA performance of agents on visual streams—namely, videos.

The 'agent' is instead represented by a VQA-inspired model, which extends popular vision-and-language paradigms to handle the added temporal dimension. We thus arrive at a multimodal world representation, produced by a model wherein the linguistic input (the question) is used as a conditioning signal for the visual processing pipeline. This method allows the system to extract and focus its computation on the visual concepts required to produce an answer.

Beyond being published as a conference paper at one of the most important computer vision conferences, VideoNavQA was offered a spotlight talk at the Visually Grounded Interaction and Language workshop held at the NeurIPS 2019 conference. This shows significant interest in the research community towards our alternative perspective to environment understanding and validates the importance of the study.

More recently, benchmarks have been proposed that tackle more complex tasks, such as interacting with objects and compositional tasks. Therefore, once we are certain about the capabilities of visually-grounded systems in controlled contexts like VideoNavQA (and finding models which achieve a closer gap to human performance), we can use the internal representations achieved by these models to construct agents which interact and make decisions in life-like environments.

Chapter 4

Hierarchical representations of structured information

4.1 Introduction and contribution overview

A crucial aspect to achieving useful world representations is structure—numerous data settings are abundant with it and many contemporary tasks tackled by machine learning can benefit from leveraging this property. For example, connectivity between entities, objects or parts is an essential characteristic of social networks (Twitter, Pinterest, citation networks), biochemical data (molecules, drug-gene interaction networks), traffic networks and even data typically processed by other models, such as images and videos. The structure in data can have various meanings—to only name a few, spatial (proximity between objects or their relative placement), causal (an event shown in a video frame influences the occurrence of another one several frames later), semantic (two groups in a social network sharing some of the aims) or temporal (the same event takes place at certain points in time in a sequence and the links exist between consecutive occurrences).

It is therefore important for a machine learning model to consider the structure present in inputs, when predicting their properties. This idea is analogous to the one leveraged in image classification, where CNNs exploit the spatial bias in visual data—this is done repeatedly, at various scales, after the image has been downsampled. We thus want GNNs to progressively coarsen the input graph; the resulting nodes would represent clusters that encompass the features of all contained nodes, which were produced by graph convolutional layers. Moreover, we need to design these models such that they eventually scale to large graphs, with millions of nodes, like the ones encountered in real-world scenarios. In the limit, social networks can reach $\mathcal{O}(\textit{world population})$ nodes!

This chapter presents two joint first-author contributions in the space of graph pooling, each of which addresses one of the challenges highlighted above. The first one introduces a sparse hierarchical graph classifier, whose performance on standard benchmarks matched the (November 2018) state-of-the-art results given by DiffPool [201]. At the same time, we reduced the memory complexity from quadratic in the number of nodes (namely, the DiffPool complexity) to linear in the number of nodes. This was achieved by combining existing approaches into a CNN-like graph classifier that alternates between convolutions and pooling steps. The main issue which we addressed here is scalability—the memory consumption plot in the experimental section shows how our method handles graphs which are too large for DiffPool to be trained on a single GPU with. I worked on the model development, ran incremental experiments (to gradually improve the performance of the pooling layer) on all datasets, ran final experiments (to match state-of-the-art results) on the Collab dataset, produced the t-SNE visualisation and wrote the the evaluation section, except for the memory usage plot and explanation.

The second contribution proposes a topologically-infused pooling operator, MPR (Mapper-based PageRank), which competed with contemporary state-of-the-art approaches on graph classification. This pooling method relies on Mapper, a topological data analysis algorithm, to incorporate useful structural information via a so-called 'lens' function. In our case, this function is based on PageRank and thus exploits the power-law distributions often encountered in real-world graphs, especially in social networks. In this case, the focus was on producing theoretically-grounded and interpretable representations—the paper appendix describes a visualisation technique also based on the Mapper algorithm, which illustrates the applicability and utility of graph summarisation methods. I developed the MPR pooling layer (while Cristian Bodnar worked on the DMP layer and visualisation), ran experiments with MPR-based models and all other baselines on all datasets and wrote the MPR pooling layer, model and evaluation sections of the paper.

We named the first study *'Towards Sparse Hierarchical Graph Classifiers'* [25] and introduced it at the Relational Representation Learning Workshop, during the Thirty-second Conference on Neural Information Processing Systems in 2018. At the time of writing this dissertation, the study had been cited 70 times.

The second contribution, '*Deep Graph Mapper: Seeing Graphs through the Neural Lens*' [16], was presented at several venues—first, at the ELLIS Workshop on Geometric and Relational Deep Learning in April 2020, then at the Topological Data Analysis Workshop, during the Thirty-fourth Conference on Neural Information Processing Systems in 2020. We have also recently submitted a manuscript to the Frontiers in Big Data journal, under the topic *Topology in Real-World Machine Learning and Data Analysis*.

4.2 Sparse differentiable pooling

Recent research in graph representation learning, mostly relying on graph convolutional networks, has brought substantial improvements on many graph-based benchmark tasks. While novel approaches to learning node embeddings at the time of the study were well suited to node classification and link prediction, their application to graph classification (predicting a single label for the entire graph) remained mostly rudimentary—often using a single global pooling step to aggregate node features or a hand-designed, fixed heuristic for hierarchical coarsening of the input graph. An essential step towards ameliorating this is differentiable graph coarsening—reducing the size of the graph in an adaptive, data-dependent manner within a graph neural network pipeline, as a generalisation of the image downsampling process within CNNs. However, the previous state-of-the-art approach to pooling had quadratic memory requirements during training and would therefore not scale to large graphs. In this work, we combined several contemporary advances in graph neural network design to demonstrate that competitive hierarchical graph classification results were possible, without sacrificing sparsity in the pipeline. Our evaluation was carried out on several established graph classification benchmarks, with results highlighting an important direction for subsequent research in learning with graph neural networks.

4.2.1 Previous related work

We studied the task of graph classification—the process of learning to place graphs into one of several classes. This can be seen as a direct generalisation of image classification [105], since images may be easily cast as a special case of a 'grid graph' (where each pixel of an image has links to its eight immediate neighbours). Therefore, we considered it a logical next step to investigate and generalise CNN-style input processing to graphs [12, 20, 73]. CNN classifiers typically comprise alternating *convolutional* and *pooling* layers, which progressively extract higher-level features from the input image, and appropriately downsample the image, respectively. The visual feature tensor eventually becomes small enough to be processed by fully-connected neural networks—the second part of the network that is responsible for prediction. Converting these architectures to handle graph-structured data is a central challenge for the field of graph representation learning, receiving substantial attention at the time of the study [12, 20, 73].

Generalising the (image) convolutional layer to graphs has been a very active research area. Lately, several graph convolutional layers [22, 42, 61, 97, 181] have been proposed, which significantly advanced the state-of-the-art on many challenging node classification benchmarks (a graph-domain task that corresponds to image segmentation), as well

as link prediction. Conversely, before this work was carried out, generalising pooling layers had received substantially smaller levels of attention by the community. One alternative approach would be to neglect the pooling operation entirely and instead apply simple aggregation or set-based neural networks to node embeddings previously generated by graph convolutional layers [37, 46, 61, 115]. Alternately, the graph may be classified through a neural network operating on the concatenation of all of its nodes; this typically requires specifying a canonical ordering over the nodes [140, 208], a substantially challenging task.

The proposed pooling methods would fall into two broad categories: (1) aggregating node representations via global pooling after each [46] or only after the last message passing step [37, 61, 115], and (2) aggregating node representations into clusters which coarsen the graph hierarchically [6, 22, 42, 51, 133, 134, 140, 164, 201]. Apart from two studies [6, 201], all earlier works in this area assumed a fixed, pre-defined cluster assignment, which was obtained by running a clustering algorithm on the graph nodes (for example, using the GraClus algorithm [45] to obtain structure-dependent cluster assignments or finding clusters via k-means on node features [134]). More recent works [6, 201] highlighted that intermediate node representations (that is, ones obtained after applying a graph convolutional layer) can be useful in obtaining both feature- and structure-based cluster assignments—these would therefore adapt to the underlying data and be learned in a differentiable fashion.

The first CNN-style graph neural network with a learnable pooling operator had been recently pioneered, leveraging the DiffPool layer [201] within an end-to-end training process. DiffPool computes soft clustering assignments of nodes from the original graph to nodes in the pooled graph. By restricting the clustering scores to the graph adjacency information and leveraging a sparsity-inducing entropy loss function, the clustering learned by DiffPool eventually converges to an almost-hard clustering with interpretable structure, and led to state-of-the-art results on several graph classification benchmarks.

The l-th DiffPool layer produces a coarsened graph $(\boldsymbol{X}_{l+1}, \boldsymbol{A}_{l+1})$ from the input one $(\boldsymbol{X}_l, \boldsymbol{A}_l)$, according to the following equations:

$$\begin{aligned} \boldsymbol{X}_{l+1} &= \boldsymbol{S}_l^\top \boldsymbol{X}_l \\ \boldsymbol{A}_{l+1} &= \boldsymbol{S}_l^\top \boldsymbol{A}_l \boldsymbol{S}_l, \end{aligned} \tag{4.1}$$

where $\boldsymbol{S}_l^\top \in \mathbb{R}^{n_l \times n_{l+1}}$ and n_k is the number of nodes in the input graph at the k-th pooling layer. The matrix $\boldsymbol{S}_l$ is a learned soft cluster assignment—S_{ij} intuitively indicates the degree of membership of the i-th node in the input graph to the j-th cluster in the coarsened graph. The cluster assignment matrix $\boldsymbol{S}_l$ is computed via a GNN-based

embedding module that takes as input the current input graph $(\boldsymbol{X}_l, \boldsymbol{A}_l)$:

$$\boldsymbol{S}_l = \text{softmax}\big(\text{GNN}_l(\boldsymbol{X}_l, \boldsymbol{A}_l)\big). \tag{4.2}$$

The output of the GNN is then passed through a softmax operator to produce probabilistic cluster assignments for each node, which satisfies the condition $\sum_{j=1}^{n_{l+1}} S_{ij} = 1, \forall i \in \{1, \ldots, n_l\}$. The first step in Equation 4.1 thus computes the features of the resulting clusters as a weighted combination of the member node features. Then, the second step in Equation 4.1 produces the adjacency matrix of the pooled graph, with edge weights indicating how strongly any two clusters are connected.

The main limitation of DiffPool is the computation of soft clustering assignments. Even though clusters eventually converge, an entire assignment matrix is stored during early training phases, to establish cluster memberships of nodes from the original graph to new nodes in the pooled graph, in an all-pairs manner. This induces a quadratic $\mathcal{O}(r|V|^2)$ memory requirement, for any pooling algorithm with a (fixed) pooling ratio r, therefore making scaling up to larger graphs more difficult for a DiffPool-based pipeline.

In this work, we leveraged contemporary advances in graph neural network research [6, 72, 195] to show that sparsity can be maintained, whilst obtaining good performance on end-to-end graph convolutional architectures with coarsening operations. We demonstrated results comparable to variants of DiffPool on four standard graph classification benchmarks, using a CNN-like graph classifier that only requires $\mathcal{O}(|V| + |E|)$ storage (equivalent to the storage complexity of the input graph).

4.2.2 A CNN-style graph classifier

Assume a standard graph machine learning setup, where the input graph is represented as a matrix of *node features* $\boldsymbol{X} \in \mathbb{R}^{N \times F}$ and an *adjacency matrix* $\boldsymbol{A} \in \mathbb{R}^{N \times N}$. I use N to denote the number of nodes in the graph, and F for the feature dimensionality. In cases where the graph has no initial node features, we can leverage the node degree information (such as the one-hot encodings of node degrees, for all degrees less or equal to a given upper bound) as artificial node features. While in general, the adjacency matrix may consist of real numbers (and may even contain edge features), here we focused on undirected and unweighted graphs—that is, $\boldsymbol{A}$ is assumed to be binary and symmetric. The complete model pipeline is depicted in Figure 4.1.

In order to build a CNN-inspired neural network for graph classification, we first define *convolutional* and *pooling* layers. In addition, we leverage a *readout* operation (which corresponds to the flattening layer in an image CNN)—the purpose of this step is to

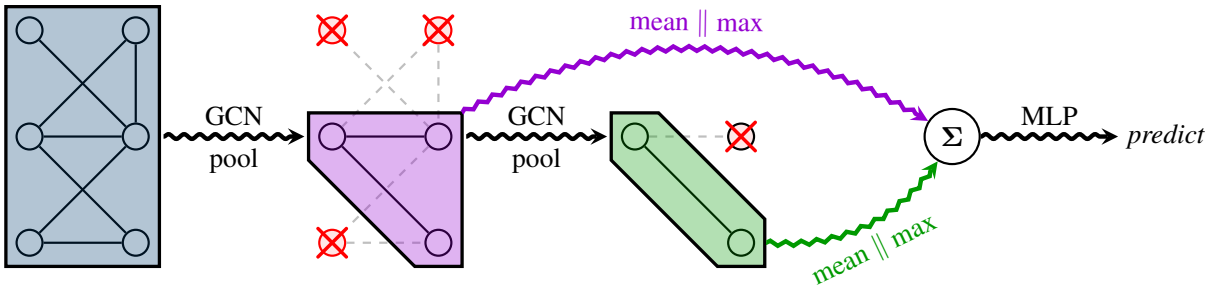

Figure 4.1: The full pipeline of our model (with $r = 0.5$), leveraging several blocks of convolutional/pooling layers. Unlike DiffPool, the pooling steps drop, rather than aggregate, nodes. The jumping-knowledge-style summary (Σ) combines information from all scales in the iterative clustering process.

convert the learned representations into a fixed-size feature vector, which in turn is used by a classifier (such as an MLP) for the final prediction. All these operations are described in the remainder of this section.

Convolutional layer Since our model is tasked with classifying unseen graph structures at test time, the main requirement of the convolutional layer is that it is inductive, which means that it should not depend on a fixed and known graph structure. A straightforward example of such a layer is the mean-pooling propagation rule, as previously used in GCN [97] or Const-GAT [181]:

$$\text{mean-pool}(\boldsymbol{X}, \boldsymbol{A}) = \sigma\left(\hat{\boldsymbol{D}}^{-1}\hat{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{\Theta} + \boldsymbol{X}\boldsymbol{\Theta}'\right), \tag{4.3}$$

where $\hat{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}_N$ is the adjacency matrix with added self-loops and $\hat{\boldsymbol{D}}$ is the corresponding degree matrix—namely, $\hat{D}_{ii} = \sum_j \hat{A}_{ij}$. We have used the rectified linear unit (ReLU) for the activation function σ. The linear transformations $\boldsymbol{\Theta}, \boldsymbol{\Theta}' \in \mathbb{R}^{F \times F'}$ are learnable and applied to each node. Finally, the transformation $\boldsymbol{\Theta}'$ corresponds to a simple skip-connection [75], which further enforces a learned preservation of information about the central node.

Pooling layer It is essential to ensure that a graph downsampling layer behaves idiomatically across a large range of graph sizes and structures. To this end, we choose to coarsen the graph with a pooling ratio $r \in (0, 1]$. This implies that pooling over a graph with N nodes will result in a new graph with $\lceil rN \rceil$ nodes.

DiffPool achieves this by computing a clustering of the N input nodes into $\lceil rN \rceil$ clusters (which incurs a quadratic penalty in storing cluster assignment scores for the backpropagation phase). Instead, we leveraged the pooling mechanism within Graph U-Nets [6], which simply drops $N - \lceil rN \rceil$ nodes from the original graph.

Within this pooling layer, the choice of nodes to drop is based on a projection score of node features $\boldsymbol{X}$ against a learnable vector $\boldsymbol{p}$. Gradients flow into $\boldsymbol{p}$ by allowing the projection scores to also be used as gating values. In this way, the retained nodes with lower scores will experience less significant feature retention in the output graph. The operation of this layer—namely, computing a pooled graph $(\boldsymbol{X}', \boldsymbol{A}')$ from an input graph $(\boldsymbol{X}, \boldsymbol{A})$—can be fully expressed as follows:

$$\begin{aligned} \boldsymbol{y} &= \frac{\boldsymbol{X}\boldsymbol{p}}{\|\boldsymbol{p}\|} \\ \boldsymbol{i} &= \text{top-}k(\boldsymbol{y}, r) \\ \boldsymbol{X}' &= (\boldsymbol{X} \odot \tanh(\boldsymbol{y}))_{\boldsymbol{i}} \\ \boldsymbol{A}' &= \boldsymbol{A}_{\boldsymbol{i},\boldsymbol{i}}. \end{aligned} \tag{4.4}$$

In the equations above, $\|\cdot\|$ is the L_2 norm, top-$k(\cdot, r)$ selects the indices corresponding to the first r% largest values in a given input vector, $\odot$ is element-wise multiplication and $(\cdot)_{\boldsymbol{i}}$ is an indexing operation that takes slices at indices specified by $\boldsymbol{i}$. The entire layer operation thus requires only point-wise projection and slicing into the original (feature and adjacency) matrices. This trivially maintains a linear memory complexity.

Readout layer Our model also requires a 'flattening' step, to preserve information about the input graph while yielding a fixed-size representation. In CNNs, the typical and natural way of achieving this is via global average pooling—here, this translates to taking the average of all learned node embeddings produced by the final graph convolutional layer. We augment this information with another summary obtained by global max pooling, as we found this strengthened our representations. Finally, inspired by the jumping-knowledge network architecture [194, 195], we perform this summarisation step after every convolutional-pooling block within the network—all of the resulting summaries are aggregated via the sum operation.

The output graph of the l-th conv-pool block, $(\boldsymbol{X}^{(l)}, \boldsymbol{A}^{(l)})$ yields a summary $\boldsymbol{s}^{(l)}$ according to the equation below:

$$\boldsymbol{s}^{(l)} = \frac{1}{N^{(l)}} \sum_{i=1}^{N^{(l)}} \boldsymbol{x}_i^{(l)} \| \max_{i=1}^{N^{(l)}} \boldsymbol{x}_i^{(l)}. \tag{4.5}$$

where $N^{(l)}$ is the number of nodes of the graph, $\boldsymbol{x}_i^{(l)}$ are the i-th node's feature vector, and $\|$ denotes concatenation. Then, the final global summary vector produced by a graph CNN with L layers is represented by the sum of all summaries (namely, $\boldsymbol{s} = \sum_{l=1}^{L} \boldsymbol{s}^{(l)}$) and fed to an MLP classifier for obtaining class predictions.

We find that aggregating information across all layers is essential—not only to preserve information at different scales, but also to handle different input graph sizes: this mechanism is able to retain information on smaller graphs that are often quickly coarsened to only a few nodes.

Model	*Enzymes*	*D&D*	*Collab*	*Proteins*
Graphlet	41.03	74.85	64.66	72.91
Shortest-path	42.32	78.86	59.10	76.43
1-WL	53.43	74.02	78.61	73.76
WL-QA	60.13	79.04	80.74	75.26
PatchySAN	–	76.27	72.60	75.00
GraphSAGE	54.25	75.42	68.25	70.48
ECC	53.50	74.10	67.79	72.65
Set2Set	60.15	78.12	71.75	74.29
SortPool	57.12	79.37	73.76	75.54
DiffPool-Det	58.33	75.47	**82.13**	75.62
DiffPool-NoLP	62.67	79.98	75.63	77.42
DiffPool	**64.23**	**81.15**	75.50	**78.10**
Ours	64.17	78.59	74.54	75.46

Table 4.1: Classification accuracy percentages. Our model successfully outperforms the sparse aggregation-based GraphSAGE baseline, while being a close competitor to DiffPool variants, across all datasets. This confirms the effectiveness of leveraging learnable pooling while preserving sparsity.

4.2.3 Experiments

Datasets and evaluation procedure We aimed to determine how well our sparse CNN-like pipeline can hierarchically compress graph representations, while still producing features relevant for classification. To achieve this, we evaluated the architecture on several well-known benchmark tasks: biological (*Enzymes*, *Proteins*, *D&D*) and scientific collaboration (*Collab*) datasets [93]. We presented the classification accuracy obtained from performing 10-fold cross-validation on each of these benchmarks, in relation to the DiffPool performance reported by Ying et al. [201].

Model parameters Our graph neural network architecture consists of three blocks, each of them containing a graph convolutional layer with 128 (Enzymes and Collab) or 64 features (D&D and Proteins), followed by a pooling layer. We ensure enough information is retained after each coarsening step by preserving 80% of the existing nodes (hence using a pooling ratio of $r = 0.8$). A learning rate of 0.005 was used

for *Proteins* and 0.0005 for all other datasets. The model was trained using the Adam optimizer [95] for 100 epochs on *Enzymes*, 40 on *Proteins*, 20 on *D&D* and 30 on *Collab*.

Results Table 4.1 illustrates our results in relation to the ones reported by [201]. In all cases, our model significantly outperforms the GraphSAGE sparse aggregation method [72], while successfully competing at most within 1% accuracy with variants of DiffPool [201], the singular development in hierarchical graph representation learning at the time our study was carried out. Unlike DiffPool, our method does not require quadratic memory, suggesting an important potential future development: deploying scalable hierarchical graph classifiers on larger, real-world inputs.

Petar Veličković also verified this claim empirically, through experiments on random inputs. As a result, Figure 4.2 demonstrates that our method compares favourably to DiffPool on larger-scale graphs, even if the pooling layer does not drop any nodes (compared to a 0.25 retain rate for the DiffPool).

Qualitative analysis Finally, I investigated the distribution of graph summaries, using a pre-trained model on a fold of the *Collab* dataset to produce 499 outputs across all 3 classes. Figure 4.3 shows that an evident clustering can be achieved, once the graph has been processed by the sequence of convolutional and pooling steps in our model.

4.2.4 Summary

We have introduced a CNN-style graph classification architecture that, at the time of publication, was the only sparse pooling-based model (memory requirement of $\mathcal{O}(|V| + |E|)$ for the pooling layer) to reach performance comparable with the state-of-the-art one reported by [201] for DiffPool. Using a combination of Top-k pooling and jumping-knowledge aggregation, which ensures that information at all levels in the hierarchy is retained, we compete with DiffPool variants within 1% accuracy across 4 standard benchmarks. This result is an essential step towards scaling up such methods to large graphs, which are often encountered in real-world data settings—here, quadratic memory requirements, like the one of DiffPool ($\mathcal{O}(|V|^2)$), become challenging and eventually infeasible.

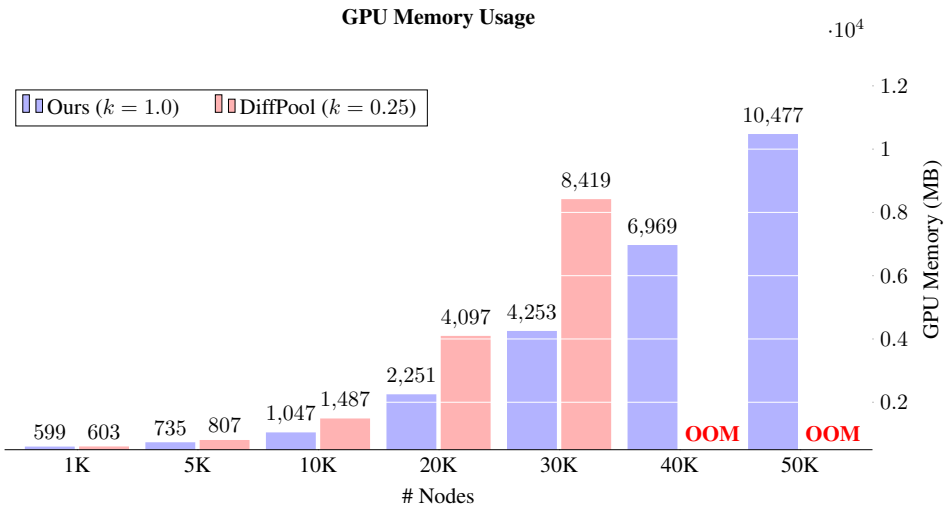

Figure 4.2: GPU memory usage of our method (with no pooling; $r = 1.0$) and DiffPool ($r = 0.25$) during training on Erdős-Rényi graphs [48] of varying node sizes (and $|E| = 2|V|$). Both methods ran with 128 input and hidden features, and three Conv-Pool layers. "OOM" denotes out-of-memory.

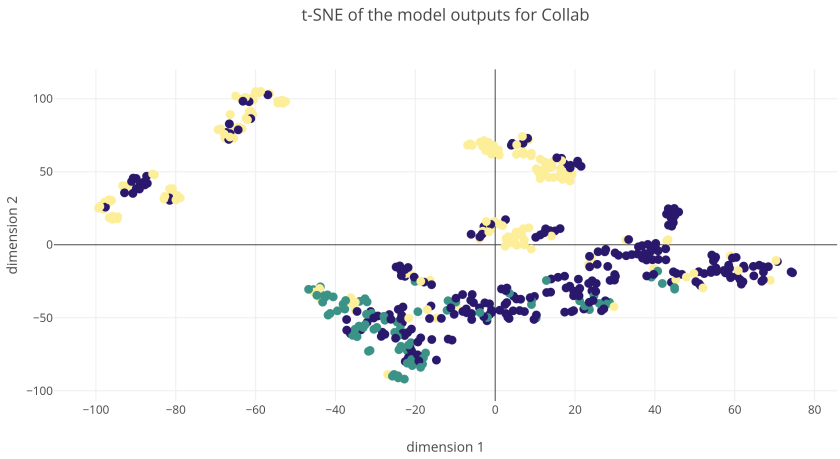

Figure 4.3: *t*-SNE plot illustrating the classification capabilities of our model. The points represent summaries of 499 *Collab* test graphs; each of the three classes corresponds to a different color.

4.3 Topologically-grounded pooling

As noted previously, the task of summarising graphs has been considerably explored in recent years. Numerous studies have been published, tackling the challenge of building pooling layers that operate on data regions with arbitrary structures. These inputs contrast the grid-like ones encountered in image processing, where fixed, non-adaptive summarisation techniques, such as max-pooling, have been empirically shown to achieve success on standard benchmarks. In our study [16], Cristian Bodnar and I have designed a general technique named Deep Graph Mapper (DGM) as a general synthesis of Mapper [71, 167], an algorithm from the field of Topological Data Analysis (TDA) [32], and graph neural networks (GNNs) [12, 20, 160]. My collaborator further showed an important connection between Mapper visualisations and graph pooling methods. Namely, he proved that Mapper is a generalisation of pooling methods based on soft cluster assignments, which include state-of-the-art algorithms like minCUT [15] and DiffPool [201]. With this idea in mind, I have proposed MPR—a novel pooling operator which merges the Mapper algorithm with GNN expressiveness to yield topologically-grounded graph summaries. Experimental studies revealed that MPR obtains competitive results with state-of-the-art methods on numerous standard benchmarks.

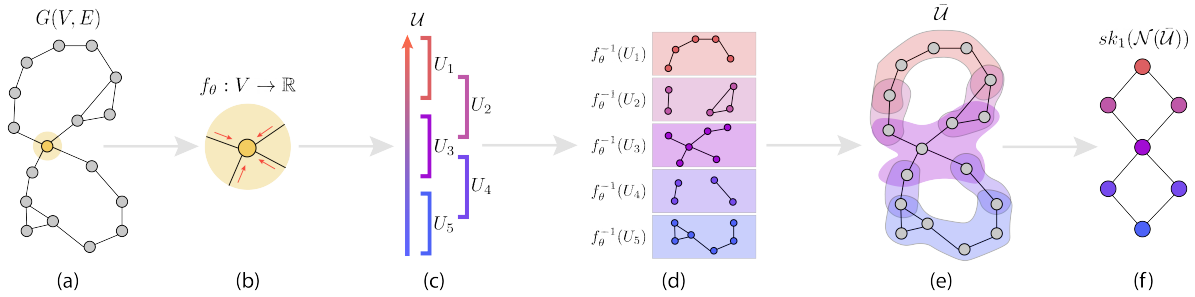

Figure 4.4: An illustration of the Deep Graph Mapper (DGM) algorithm. For simplicity, a graph neural network (GNN) is tasked with approximating a 'height' function over the nodes, as viewed above (for example, the nodes in the lower triangle yield smaller function values, while the ones in the upper triangle result in higher ones). The input graph seen in subfigure (a) is passed through the GNN. The latter is applied to each of the nodes in the graph (b) and outputs a real number (the approximation of the node 'height') (c). Given a cover $\mathcal{U}$ of the image of the GNN (c), the refined pullback cover $\bar{\mathcal{U}}$ is computed (d). This allows each node to be assigned to one or more clusters (e). The 1-skeleton of the nerve of the pullback cover provides the visual summary of the graph (f). The diagram is inspired by Hajij et al. [71].

Building upon the topological perspective of graph pooling, we have each proposed a graph coarsening algorithm based on Mapper. The remainder of this chapter focuses on the one I designed, MPR, mentioned in the previous paragraph. MPR leverages a fixed PageRank-based lens function and achieves results competitive with other state-of-the-

art pooling methods on graph classification benchmarks. Most notably, the pooling layer competes with the differentiable one designed by Cristian Bodnar—this result showcases the power of well-chosen inductive biases reflected in the node scoring functions, given a certain data distribution in the experimental setup.

4.3.1 Background and relevant work

Graph coarsening methods have been extensively explored as part of GNN graph classification frameworks. Luzhnica et al. [118] proposed a topological approach to pooling, progressively summarising the graph by aggregating its maximal cliques into new clusters. However, cliques represent local topological features—in contrast, our methods employ a global perspective of the graph during the pooling step. Two main paradigms have emerged in the design of adaptive coarsening layers: Top-k pooling, based on a learnable ranking [57], and learning the cluster assignment [201] with additional entropy and link prediction losses for more stable training (DiffPool). Following either of these directions, several variants and incremental improvements have been proposed. The Top-k approach has been combined with jumping-knowledge networks [25], attention [86, 111] and self-attention for cluster assignment [151]. In a manner similar to DiffPool, the method suggested by Bianchi et al. [15] uses several loss terms to encourage the formation of clusters with strongly connected nodes, similar sizes and orthogonal assignments. An alternative approach is suggested by Ma et al. [120], which leverages spectral clustering.

I will now review the Mapper [167] algorithm, with a focus on graph domains [71].

Definition 4.3.1. Assume X is a topological space, $f : X \to \mathbb{R}^d, d \geq 1$ is a continuous function and $\mathcal{U} = (U_i)_{i \in I}$ is a cover of $\mathbb{R}^d$. The **pullback cover** $f^*(\mathcal{U})$ of X induced by $(f, \mathcal{U})$ is therefore the collection of open sets $f^{-1}(U_i), i \in I$, for some indexing set I. The cover of X, which is formed by the connected components of all pre-images in the pullback cover $f^*(\mathcal{U})$, is called the **refined pullback cover**, denoted by $\bar{\mathcal{U}} = (\bar{U}_j)_{j \in J}$, where J is an indexing set.

Definition 4.3.2. Assume X is a topological space with an open cover $\mathcal{U} = (U_i)_{i \in I}$. The **1-skeleton of the nerve** $\mathcal{N}(\mathcal{U})$ of $\mathcal{U}$, denoted by $sk_1(\mathcal{N}(\mathcal{U}))$, is the graph with nodes represented by $(v_i)_{i \in I}$, where an edge exists between two nodes v_i, v_j if and only if $U_i \cap U_j \neq \emptyset$.

Given a graph $G = (V, E)$, a carefully designed **lens function** $f : V \to \mathbb{R}^d$ and cover $\mathcal{U}$ of $\mathbb{R}^d$, the Mapper algorithm first computes the associated pullback cover $f^*(\mathcal{U})$. Then, using a clustering algorithm of choice, it processes each of the sets of vertices $f^{-1}(U_i)$ in $f^*(\mathcal{U})$ and obtains the refined pullback cover. During the last step, Mapper outputs

a summarised graph by computing the 1-skeleton of the nerve of the refined pullback cover, $sk_1(\mathcal{N}(\bar{\mathcal{U}}))$. At a high level, the soft clusters formed by the refined pullback become the nodes of the output graph, with edges connecting any two nodes that correspond to overlapping clusters.

4.3.2 A Mapper-based coarsening layer

I considered a fixed and scalable non-differentiable lens function $f : V \to \mathbb{R}$ that is given by the normalised PageRank (PR) [144] value for a given node. The PageRank function computes an importance score for each node based on its connectivity, according to the popular recurrence relation:

$$f(\boldsymbol{X}_i) \triangleq \mathbf{PR}_i = \sum_{j \in \mathcal{N}(i)} \frac{\mathbf{PR}_j}{|\mathcal{N}(i)|}, \tag{4.6}$$

where $\mathcal{N}(i)$ represents the set of neighbours of the i-th node in the graph. The resulting scores are in the $[0, 1]$ range, thereby reflecting the probability of a random walk through the graph to end in the corresponding node. Using the previously described cover $\mathcal{U}$ (which is essentially the set of overlapping intervals), the elements of the pullback cover form a soft cluster assignment matrix $\boldsymbol{S}$:

$$S_{ij} = \frac{\mathbb{I}_{i \in f^{-1}(U_j)}}{|\{U_k | i \in f^{-1}(U_k)\}|} \tag{4.7}$$

where U_n is the n-th overlapping interval in the cover $\mathcal{U}$ of $[0, 1]$. A useful observation at this stage is that every resulting cluster contains nodes that have been assigned similar PageRank values. Intuitively, this pooling layer exploits the power-law distributions abundant in social data settings: it merges the (generally few) highly connected nodes in the graph, while at the same time clustering the (usually numerous) dangling nodes, which have a very small normalised PageRank score (close to zero). This method thereby favours the information that is present in the most 'important' nodes of the graph.

Since $\mathbf{PR}$ is the principal eigenvector of the transition matrix $\mathcal{M}$ of the graph, I computed the $\mathbf{PR}$ vector via power iteration, using NetworkX [70]:

$$\mathbf{PR} = \big(\alpha\mathcal{M} + (1 - \alpha)\frac{1}{N}\boldsymbol{E}\big)\mathbf{PR}, \tag{4.8}$$

where $\boldsymbol{E}$ is a matrix with all elements equal to 1 and $\alpha \in [0, 1]$ is the probability of continuing the random walk at each step. A value of α closer to 0 suggests that nodes would be ranked more uniformly and tend to form a single cluster. I employed the widely-adopted $\alpha = 0.85$; Boldi et al. [18] provide more details on PageRank.

Finally, I used the mapping $\boldsymbol{S}$ to compute $\boldsymbol{X}_{\text{MG}}$—features for the new nodes (namely, the soft clusters formed by the pullback)—and the corresponding adjacency matrix, $\boldsymbol{A}_{\text{MG}}$, as follows:

$$\begin{aligned} \boldsymbol{A}_{\text{MG}} &= \boldsymbol{S}^\top \boldsymbol{A} \boldsymbol{S} \\ \boldsymbol{X}_{\text{MG}} &= \boldsymbol{S}^\top \boldsymbol{X}. \end{aligned} \tag{4.9}$$

It is crucial that graph classification models are permutation-invariant with respect to node ordering, since a graph $(\boldsymbol{X}, \boldsymbol{A})$ can be represented by any tuple $(\boldsymbol{X}\boldsymbol{\Pi}, \boldsymbol{A}\boldsymbol{\Pi})$, where $\boldsymbol{\Pi}$ is a node permutation. Below, I show that the MPR pooling method has this property.

Proposition 4.3.1. The PageRank pooling operator defined above is permutation-invariant.

Proof. First, I note that the PageRank function is permutation-invariant and refer the reader to Altman [1, Axiom 3.1] for the proof. It then trivially follows that the PageRank pooling operator is permutation-invariant. □

4.3.3 Graph classification model

In a graph classification task, each sample G is represented by a tuple $(\boldsymbol{X}, \boldsymbol{A})$, where $\boldsymbol{X}$ is the node feature matrix and $\boldsymbol{A}$ is the adjacency matrix. Both graph embedding and classification networks comprise a sequence of graph convolutional (GCN) layers [97], with the l-th layer processing the input feature matrix as follows:

$$\boldsymbol{X}_{l+1} = \sigma(\hat{\boldsymbol{D}}^{-\frac{1}{2}} \hat{\boldsymbol{A}} \hat{\boldsymbol{D}}^{-\frac{1}{2}} \boldsymbol{X}_l \boldsymbol{W}_l), \tag{4.10}$$

where $\hat{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$ is the adjacency matrix with self-loops, $\hat{\boldsymbol{D}}$ is the normalised node degree matrix and σ is the activation function. After E layers, the **embedding network** outputs the node embeddings $\boldsymbol{X}_{L_E}$, which are then processed by a pooling layer to coarsen the graph. The classification network first takes as input node features of the Mapper-pooled graph[1] $\boldsymbol{X}_{\text{MG}}$, as shown in Figure 4.4. These features are then passed through L_C graph convolutional layers. Finally, the **classification network** computes a graph summary given by the feature-wise node average and applies a final linear layer which predicts the class:

$$\hat{\boldsymbol{y}} = \text{softmax}\Big(\frac{1}{|\text{MG}|} \sum_{i=1}^{|\text{MG}|} \boldsymbol{X}_{L_C} \boldsymbol{W}_{\text{f}} + \boldsymbol{b}_{\text{f}}\Big). \tag{4.11}$$

[1] In the general case, one or more {embedding → pooling} operations are performed sequentially within the pipeline.

4.3.4 Experiments

Tasks and setup We have motivated the suitability of the Mapper-GNN synthesis within a pooling framework by evaluating DMP (the differentiable pooling layer that Cristian Bodnar developed) and MPR in a variety of data settings: social (IMDB-Binary, IMDB-Multi, Reddit-Binary, Reddit-Multi-5k), citation networks (Collab) and chemical data (D&D, Mutag, NCI1, Proteins) [93]. We employ 10-fold cross-validation across all tasks to evaluate the graph classification performance of DMP, MPR and other competitive state-of-the-art methods. The random seed was set to zero for all experiments (with respect to dataset splitting, shuffling and parameter initialisation), such that we obtained a fair comparison across all architectures. The models were trained on a single Titan Xp GPU, using the Adam optimiser [95] with early stopping on the validation set, for a maximum of 30 epochs. We report the classification accuracy using 95% confidence intervals calculated for a population size of 10 (the number of folds).

Baseline and model details I compared the performance of MPR to two other pooling methods that Cristian Bodnar identified mathematical connections with: minCUT [15] and DiffPool [201]. Additionally, I included Graph U-Net [57] in our evaluation, as it has been shown to yield competitive results while performing pooling from the perspective of a learnable node ranking. I denote this approach by Top-k in the remainder of the section. The non-pooling baselines also present in the evaluation are the WL kernel [162], a 'flat' model (2 message-passing steps, followed by global average pooling) and an average-readout linear classifier.

I performed hyperparameter search on MPR (see Appendix A.3) with respect to the cover cardinality n, the cover overlap (percentage g), learning rate and hidden size. The Top-k architecture was evaluated using the code provided in the official repository, where separate configurations are defined for each of the benchmarks. The minCUT architecture is represented by the sequence of operations described by Bianchi et al. [15]: *MP(32)-pooling-MP(32)-pooling-MP(32)-GlobalAvgPool*, followed by a linear softmax classifier. The *MP(32)* block represents a message-passing operation performed by a graph convolutional layer with 32 hidden units $\boldsymbol{X}^{(t+1)} = \text{ReLU}(\tilde{\boldsymbol{A}}\boldsymbol{X}^{(t)}\boldsymbol{W}_m + \boldsymbol{X}^{(t)}\boldsymbol{W}_s)$, where $\tilde{\boldsymbol{A}} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$ is the symmetrically-normalised adjacency matrix and $\boldsymbol{W}_m, \boldsymbol{W}_s$ are learnable weight matrices representing the message passing and skip-connection operations within the layer. The DiffPool model follows the same sequence of steps.

The optimised architectures have the following configurations:

- MPR—learning rate $5e^{-4}$, hidden sizes $\{128, 128\}$ and:
  - D&D and Collab: cover sizes $\{20, 5\}$, interval overlap 10%, batch size 32;

  - Proteins: cover sizes $\{8, 2\}$, interval overlap 25%, batch size 128;
  - Reddit-Binary: cover sizes $\{20, 5\}$, interval overlap 25%, batch size 32;

- Top-k—specific dataset configurations, as provided in the official GitHub repository;
- minCUT—learning rate $1e^{-3}$, same architecture as reported by the authors in the original work [15];
- DiffPool—learning rate $1e^{-3}$, hidden size 32, two pooling steps, pooling ratio $r = 0.1$, global average mean readout layer, with the exception of Collab and Reddit-Binary, where the hidden size was 128.

Model	D&D	Mutag	NCI1	Proteins	Collab	IMDB-B	IMDB-M	Reddit-B	Reddit-5k
Top-k	75.1 ± 2.2	82.5 ± 6.8	67.9 ± 2.3	74.8 ± 3.0	75.0 ± 1.1	69.6 ± 3.8	45.0 ± 2.8	79.4 ± 7.4	48.5 ± 1.1
minCUT	77.6 ± 3.1	82.9 ± 6.0	68.8 ± 2.1	73.5 ± 2.9	79.9 ± 0.8	70.7 ± 3.5	50.6 ± 2.1	$\mathbf{87.2 \pm 5.0}$	$\mathbf{52.9 \pm 1.3}$
DiffPool	$\underline{77.9 \pm 2.4}$	$\mathbf{94.7 \pm 7.1}$	68.1 ± 2.1	74.2 ± 0.3	81.3 ± 0.1	72.4 ± 3.1	50.3 ± 1.8	79.0 ± 1.1	50.4 ± 1.7
WL	77.4 ± 2.6	74.5 ± 6.5	$\mathbf{76.4 \pm 2.7}$	74.7 ± 3.2	78.5 ± 1.1	72.1 ± 3.1	$\underline{50.7 \pm 2.9}$	66.7 ± 10.4	49.2 ± 1.4
Flat	69.9 ± 2.2	71.8 ± 4.3	65.5 ± 1.7	70.2 ± 2.6	80.9 ± 1.4	$\underline{73.6 \pm 4.2}$	48.5 ± 2.4	70.0 ± 10.8	49.5 ± 1.7
avg-MLP	63.7 ± 1.4	69.1 ± 5.8	55.7 ± 2.8	61.8 ± 1.7	74.8 ± 1.3	71.5 ± 2.9	49.5 ± 2.2	53.6 ± 6.2	45.9 ± 1.6
DMP (Ours)	77.3 ± 3.6	$\underline{84.0 \pm 8.6}$	$\underline{70.4 \pm 4.2}$	$\mathbf{75.3 \pm 3.3}$	$\underline{81.4 \pm 1.2}$	$\mathbf{73.8 \pm 4.5}$	$\mathbf{50.9 \pm 2.5}$	86.2 ± 6.8	51.9 ± 2.1
MPR (Ours)	$\mathbf{78.2 \pm 3.4}$	80.3 ± 6.0	69.8 ± 1.8	$\underline{75.2 \pm 2.2}$	$\mathbf{81.5 \pm 1.0}$	73.4 ± 2.7	50.6 ± 2.0	$\underline{86.3 \pm 4.8}$	$\underline{52.3 \pm 1.6}$

Table 4.2: Results obtained on classification benchmarks. Accuracy measures with 95% confidence intervals are reported. The highest result is **bolded** and the second highest is underlined. The first four columns correspond to molecular graphs, while the others, to social graphs. Our models (DMP, MPR) perform competitively with other state-of-the-art methods.

Results I present the graph classification performance obtained by all models in Table 4.2. The reported results show that on the social domain, MPR ranked either first or second, or achieved accuracy scores within 0.5% of the best-performing model. This finding confirms the hypothesis that the PageRank-based pooling layer exploits the power-law distributions present in social data. Meanwhile, the performance of DMP was found to be similar on social data and generally higher on molecular graphs. We attributed this to the fact that all nodes in molecular graphs tend to have a similar PageRank score—MPR is therefore likely to assign all nodes to one cluster, effectively performing a readout. In this domain, DMP performs particularly well on Mutag, where it is second-best and improves by 3.7% over MPR, showing the benefits of having a differentiable lens in challenging data settings.

Overall, MPR achieves the best accuracy on 2 datasets (D&D, Collab) and the next best result on 3 more (Proteins, Reddit-Binary and Reddit-Multi-5k). DMP improves on MPR by less than 1% on NCI1, Proteins, IDMB-Binary and IMDB-Multi, showing the perhaps surprising strength of the simple, fixed-lens pooling operator.

4.3.5 Summary

In this section, I have introduced the MPR pooling layer, a topologically-grounded method for coarsening graphs with the help of GNNs, that is readily applicable in a graph classification setting. The appendix of the original manuscript contains a proof that Mapper is a generalisation of soft cluster assignment methods, thereby providing a bridge between graph pooling and the TDA literature. Based on this connection, we have proposed two Mapper-based pooling operators: a non-differentiable one that scores nodes using PageRank and exploits power-law distributions in the data, designed by me, and a differentiable one that uses RBF kernels to simulate the cover, which Cristian Bodnar worked on. Our experiments have shown that both layers yield architectures competitive with several state-of-the-art methods on graph classification benchmarks.

Notably, the MPR pooling layer achieves higher performance than the differentiable DMP layer on 4 benchmarks, including one from the biochemical domain! I view this result as a promising start to designing pooling operators which contain domain knowledge. The latter can be minimally present in the form of an inductive bias over the data distribution contained in the scoring function—similarly to PageRank being suitable for social settings.

4.4 Discussion and summary

This chapter has introduced two approaches to graph pooling, while discussing coarsening methods through the prism of producing hierarchical graph representations for property prediction. The main motivation for studying these approaches is the importance of progressively downsampling graphs, instead of simply aggregating all nodes at once. This view is supported by several arguments—firstly, the structure or semantics of the node features may comprise important discriminative signals themselves, so it is almost always useful to consider connectivity in addition to the independent node samples. Moreover, graph convolutions alone cannot encompass the entire graph structure, but only local node neighbourhoods, so simply applying a sequence of such layers is likely to discard useful intermediate-scale information present in the input graph. In the limit, we would apply enough operations to cover the entire graph depth—however, it is currently challenging to achieve meaningful representations in this way. Issues like feature 'washout' are challenging to address—Zhang [207] has only recently proposed a method that showed competitive results when scaling up to tens of layers.

At the time of publication, the first study was the first one to reach the ballpark of state-of-the-art performance on standard benchmarks via a sparse pooling mechanism—the authors of Top-k pooling (and implicitly Graph U-Nets [57]) only later reported graph

classification results on their OpenReview ICLR submission [6]. This was an essential step in achieving hierarchical representations for large graphs, within a CNN-style classification pipeline. The research community agrees with the importance of our study, as the paper has received 70 citations so far.

Complementary to the first contribution, the second study investigated principled and theoretically-grounded graph pooling methods, which explicitly make use of the input topology. Several GNN models only make limited use of the topological information available [43, 103]; more recently, Zhao et al. [209] have developed a persistent homology-based GNN approach to node classification, but the structural information is still limited to local node neighbourhoods. In our work, graph summaries are computed by incorporating the Mapper TDA algorithm into the pooling layer. The latter receives as input intermediate node features from the GNN and applies a so-called 'lens' function—herein, nodes with similar topological properties (for example, if one uses PageRank as the 'lens', their level of connectivity is taken into account) are eventually clustered together and a condensed representation of the graph is output by the pooling layer. Essentially a skeleton of the original graph, this new, smaller graph encompasses all the relevant information via aggregating semantically-similar nodes, while keeping the initial structural connectivity. In the future, countless 'lens' functions can be designed, depending on the domain properties—this is the aspect which makes our method powerful and general. We hope that both theoreticians and practitioners will investigate new pooling functions, pushing state-of-the-art performance further on existing benchmarks and hopefully newly proposed ones on challenging domains. A further relevant direction would be devising adaptive topological pooling for graphs that evolve over time, as these are encountered in a range of real-world scenarios, with prime examples including traffic, drug-gene interaction and social networks.

Chapter 5

Structural biases for probabilistic modelling in challenging scenarios

5.1 Introduction and contribution overview

Sometimes, learning world representations might need to be carried out under challenging conditions. Labelled data is often scarce in real-world setups, unlike in typical machine learning research experimental ones, where thousands and even millions of samples are available during training. An even bigger challenge is the underlying data distribution—it is reasonable to expect that once an ML system is deployed in practice, it will need to make predictions for out-of-distribution (OOD) samples (that is, data points that do not come from the initial distribution which the model has been trained on). However, the research literature has extensively presented cases where state-of-the-art approaches fail on OOD data points. Crucially, real-world tasks similar to the ones humans excel at often require few-shot learning (few labelled samples) or even zero-shot learning (no labelled samples)—namely, achieving the same level of performance on a variety of (possibly related) tasks, all while using the same system. This argument ties in with the previous aspect about having access to only a few labels, but also emphasises the challenge brought by the machine learning meta-task itself: adapting on-the-fly to a new setting where the data has its own, novel complexities which the system needs to grasp, model and output predictions about.

Gaussian processes are a class of probabilistic models that have been designed to tackle some of these challenges. They are able to represent stochastic processes that generate functions, but come with their own set of limitations, which include being more time-consuming to train, especially when more expressive and powerful variants, such as deep Gaussian processes, are being used. More recently, Neural Processes (NPs) were pro-

posed as a favourable trade-off between neural networks and Gaussian processes. This encoder-decoder class of models is capable of linear-time predictions ($\mathcal{O}(num\_samples)$) like neural networks, at the same time modelling functional distributions similarly to Gaussian processes, with demonstrated competitive performance. Impressively, NPs achieve this by only using a limited set of labelled points (*context set*) in a dataset (namely, a set of points sampled from a function f from the underlying distribution). The NP predicts the (high-dimensional) mean and variance of f, which is then used to make predictions for the rest of the dataset (*target set*).

As extensively discussed already in previous chapters, it is essential to model structure in world representations, whenever this information is available. However, no existing NP model leverages the relational structure between data samples for classification, with a single model tackling edge imputation (a somewhat related task, but with entirely different semantics—*predict the presence or something about the interaction between two entities* vs. *predict a quantity of interest about a single entity*). My collaborators—Ben Day, joint-first author, and Arian Jamasb, second author—and I therefore contributed to the Neural Process family by adding structural inductive biases to NPs. This type of development had been previously acknowledged as an important one within this class of models [66]. The bias is incorporated in the encoder and decoder components of the Neural Process via Message Passing, a general paradigm within the graph representation learning research community.

We have submitted the resulting paper, *'Message Passing Neural Processes'*, to the Thirty-eighth International Conference on Machine Learning. Ben Day and I have also presented this work during the Artificial Intelligence Research Talk series, with considerable exposure outside the department, and I also introduced it during the *Graph Neural Networks* module that is part of our departmental Master's course *'Advanced Topics in Machine Learning or Natural Language Processing'*, in Lent term 2021. As a joint first-author on this paper, I worked on model design, ran experiments on the ShapeNet and TUD datasets, produced the uncertainty plots, ran the active learning experiment and wrote the introduction, model, related work sections of the paper, along with the derivations and model details found in the appendix.

5.2 Incorporating relational inductive biases in the neural process model

Reviewed in Section 2.1.8, Neural Processes (NPs) are a class of models which incorporate uncertainty in building representations of stochastic processes, while maintaining a linear time complexity. Even though their characteristics render them powerful and

flexible, NPs produce a latent description by aggregating representations of context points that have been computed independently, thus lacking the ability to exploit the relational information abundant in many datasets. This hinders NPs in scenarios where the stochastic process is primarily governed by rules over the neighbourhood of a data point—such as cellular automata (CA)—and is likely to limit model performance on tasks where relational information is available. To address this shortcoming, we introduced Message Passing Neural Processes (MPNPs), the first class of NPs to explicitly leverage relational structure within the framework. Our extensive evaluation on existing benchmarks and novel CA and Cora-Branched tasks showed that MPNPs excel in relation to NPs, thriving at lower sampling rates. We further reported strong generalisation over density-based CA sets of rules and significant improvements in challenging arbitrary-labelling and few-shot learning settings.

I explain the motivation behind our work, starting with a discussion on **neural networks** (NNs). These models are widely adopted in single-task scenarios, especially where large quantities of labelled data are typically available. NNs have favourable properties such as $\mathcal{O}(|D|)$ prediction time complexity, where D is the set of samples. On the other hand, it is often difficult to adapt them to challenging scenarios—such as multi-task or few-shot learning—and uncertainty estimates for predictions are not commonly provided. **Relational inductive biases** have been more recently added to NNs [12], which resulted in a class of models called Graph Neural Networks able to exploit relational information via message-passing operations. Alternately, **Gaussian Processes** (GPs) [187] are better suited to non-standard tasks and compute uncertainty estimates, albeit at prediction costs ($\mathcal{O}(|D|^3)$) that often do not scale to real-world datasets.

Garnelo et al. [59] introduced **Neural Processes** (NPs) as a means of combining the best of both worlds. NPs learn to represent a stochastic process using labelled samples from its instantiations, with a global latent variable modelling the stochasticity of the learned functions. At test time, only a few labelled points are required to produce predictions for the rest of the dataset, along with their associated uncertainties, in linear time. These models have achieved favourable results in few-shot learning and multi-task setups [58, 59, 156], but do not leverage the structural information in the data, an approach which has been rendered highly effective on relational tasks [210].

The main contribution of our study [41] is a novel Neural Process framework for classification, which explicitly incorporates structural information when modelling stochastic processes. In this way, our modifications parallel those present in the Convolutional Conditional Neural Process (ConvCNP) framework [66], which also equips NP models with a stronger, relevant inductive bias that helps build richer functional representations. The **Message Passing Neural Process** (MPNP) is, to the best of our knowledge, the first

node classification framework that learns to represent stochastic processes which yield datasets with explicit relational information.

We have validated the relative strengths of MPNPs via experiments on a variety of existing geometric and biological tasks. Furthermore, Ben Day designed a challenging new collection of Cellular Automata datasets, which test the ability of the model to handle large variations in the function distribution. Here, it is expected that most existing baselines fail to achieve better-than-chance results. He also constructed *Cora-Branched*—a set of novel arbitrary-labelling and few-shot learning tasks derived from the Cora dataset—and again showed significant MPNP gains. In addition, Arian Jamasb worked on setting up the PPISP biochemical task and comparing the MPNP against other baselines. Throughout the chapter, I will focus on describing my contributions to this work—however, I will also mention the new tasks and results on them, if this helps the reader achieve a clear understanding of MPNP capabilities.

5.3 Previous related work

5.3.1 Neural process models

Neural Processes were proposed by Garnelo et al. [59] as a beneficial fusion between neural networks and Gaussian Processes. *Conditional Neural Processes* (CNPs) [58] are NP variants that do not model a global latent variable—this results in a deterministic dependence on the context set when making predictions. *Attentive NPs* [94], *CNAPs* [156], *Convolutional CNPs* [66] and *Sequential NPs* [166] modify the initial NP framework to reduce underfitting, better adapt to multi-task settings, and introduce stronger translational and temporal inductive biases, respectively. Louizos et al. [117] propose the *Functional NP*, which learns a graph of dependencies between the latent representations of points, without placing a prior over the latent global variable. However, the tasks they study do not contain explicit relational information. Most closely related to the MPNP is the *Graph NP* [29], which performs edge imputation using a CNP-based model and Laplacian-derived features for the context set. However, despite the semantically-similar names, Graph NPs and MPNPs tackle different classes of tasks—the former was evaluated on link prediction, which is not in the scope of our work. Our NP-based model is additionally more flexible, yields uncertainty estimates and learns from local node neighbourhoods, rather than whole-graph features, aiming to classify individual dataset samples (nodes), while leveraging the structure between them (edges).

5.3.2 Graph learning under uncertainty

Graph Gaussian Processes [137] were built starting from GPs—both the covariance function and prior exploit the features that are present in node neighbourhoods. Graph GPs are the only Gaussian method for node classification, but achieve slightly worse results than GCNs—a subclass of GNNs that we have used as a baseline in our experiments. Moreover, the complexity of Graph GPs is higher: $\mathcal{O}(\texttt{max\_node\_degree}^2 * N)$ vs. $\mathcal{O}(N)$ for (MP)NP, where $N =$ set of observations/context nodes. The *Relational GP* [35] was designed to model pairwise undirected links between data points and thus tackles a different task. The *Graph Convolutional GP* [184] is a translation-invariant model that operates in a manner similar to convolutional layers, but generalises to non-Euclidean domains. Opolka and Liò [143] have also recently proposed a *Graph Convolutional GP* model for link prediction, wherein a GP produces node-level predictions, another GP builds on the first one to obtain edge-level predictions, and a deep GP incorporates these building blocks to yield representations that are more expressive.

5.4 Message Passing Neural Processes

I present Message Passing Neural Processes (MPNPs) as the synthesis of the MP and NP models. Figure 5.1 illustrates the operation of an MPNP. I describe each step below, with Algorithm 1 summarising the MPNP label generation process.

5.4.1 Problem statement

Given a set of *nodes* that are partially labelled, with corresponding features $\boldsymbol{X}$ *and neighbours given by the adjacency matrix* $\boldsymbol{A}$, sampled from an underlying function $f : X, A \to Y$, with $f \sim \mathcal{D}$, the goal is to predict labels for a subset of the unlabelled *nodes*.

5.4.2 Dataset sampling

In a classification setup, let the context set of a dataset (here, the dataset corresponds to a graph) be defined as a set $\mathcal{C} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$ containing tuples of nodes and their respective (one-hot) labels. The encoder h is thus provided with information about the dataset in the form of a set $\mathcal{C} \cup \{\boldsymbol{x}_j \mid j \in \bigcup_{i \in \text{context set}} \mathcal{N}(i)\}$, with $|\mathcal{C}| = m$. This set contains the context set *and* the k-hop neighbourhoods of all context nodes. In this way, the MPNP leverages the relational structure between the nodes in the context set and other nodes to yield richer representations of the context nodes. In turn, this enables the global latent variable $\boldsymbol{z}$ to encode the relational structure generated by the underlying stochastic process. The target set, $\mathcal{T} = \{\boldsymbol{x}_i \mid i \in \text{context set}\} \cup \{\boldsymbol{x}_i \mid i \notin \text{context set}\}$, with

Algorithm 1: MPNP label prediction (generation). The context set is passed through the encoder, producing individual representations that are aggregated to parameterise the global latent variable. Decoding starts by sampling the latent variable and processing it jointly with each of the target points, to predict corresponding labels and associated uncertainties.

Input : Context set $\mathcal{C} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}$, with $|\mathcal{C}| = m$, features of context set node neighbours $\{\boldsymbol{x}_i \parallel j \in \bigcup_{i \in \text{context set}} \mathcal{N}(i)\}$, target set $\mathcal{T} = \{\boldsymbol{x}_i \mid i \in \text{context set}\} \cup \{\boldsymbol{x}_i \mid i \notin \text{context set}\}$, with $|\mathcal{T}| = n > m$, features of target set node neighbours $\{\boldsymbol{x}_i \parallel j \in \bigcup_{i \in \text{target set}} \mathcal{N}(i)\}$.

Output : Target label predictions $\{\hat{\boldsymbol{y}}_i \parallel i \in \text{target set}\}$.

`// Initialise node features`

foreach $i \in$ *context set* **do**
  $\boldsymbol{h}_i^0 \leftarrow \boldsymbol{x}_i \parallel \boldsymbol{y}_i$
  foreach $j \in \bigcup_{i \in \textit{context set}} \mathcal{N}(i)$ **do**
    $\boldsymbol{h}_j^0 \leftarrow \boldsymbol{x}_j \parallel \boldsymbol{0}$

`// Encoding`

foreach $i \in$ *context set* **do**
  $\boldsymbol{h}_i^0 \leftarrow \text{ReLU}(L_1(\boldsymbol{h}_i^0))$
  foreach $j \in \bigcup_{i \in \textit{context set}} \mathcal{N}(i)$ **do**
    $\boldsymbol{h}_j^0 \leftarrow \text{ReLU}(L_1(\boldsymbol{h}_j^0))$

foreach $t \in 1, ..., T$ **do**
  foreach $i \in$ *context set* **do**
    $\boldsymbol{h}_i^t \leftarrow MP(\boldsymbol{h}^{t-1})$

foreach $i \in$ *context set* **do**
  $\boldsymbol{r}_i \leftarrow L_2(\boldsymbol{h}_i^T)$

`// Aggregation`

$\boldsymbol{r} \leftarrow a(\{\boldsymbol{r}_i \parallel i \in \text{context set}\})$

`// Decoding`

Sample $\boldsymbol{z}' \sim \mathcal{N}(\mu(\boldsymbol{r}), \text{diag}[\sigma(\boldsymbol{r})])$

foreach $i \in$ *target set* **do**
  $\boldsymbol{h}_i'^0 = \boldsymbol{x}_i \parallel \boldsymbol{z}'$
  foreach $j \in \bigcup_{i \in \textit{target set}} \mathcal{N}(i)$ **do**
    $\boldsymbol{h}_j'^0 \leftarrow \boldsymbol{x}_j \parallel \boldsymbol{z}'$

foreach $i \in$ *target set* **do**
  $\boldsymbol{h}_i'^0 \leftarrow \text{ReLU}(L_1(\boldsymbol{h}_i'^0))$
  foreach $j \in \bigcup_{i \in \textit{target set}} \mathcal{N}(i)$ **do**
    $\boldsymbol{h}_j'^0 \leftarrow \text{ReLU}(L_1(\boldsymbol{h}_j'^0))$

foreach $t \in 1, ..., T$ **do**
  foreach $i \in$ *target set* **do**
    $\boldsymbol{h}_i'^t \leftarrow MP(\boldsymbol{h}'^{t-1})$

foreach $i \in$ *target set* **do**
  $\boldsymbol{r}_i' \leftarrow \text{ReLU}(L_2(\boldsymbol{h}_i'^T))$
  $\hat{\boldsymbol{y}}_i' \sim \mathcal{N}\Big(\text{softmax}(\mu(\boldsymbol{r}_i')), \text{diag}[\big(0.1 + 0.9 \times \text{softplus}(\sigma(\boldsymbol{r}_i'))\big)]\Big)$

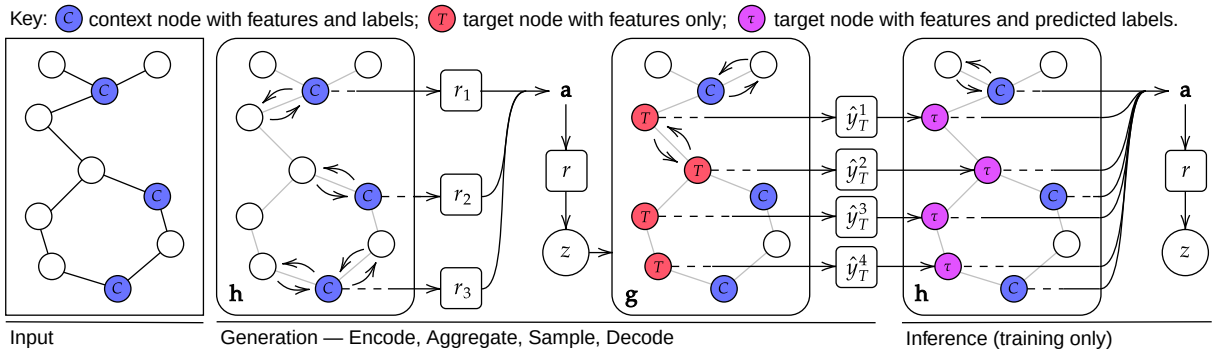

Figure 5.1: Computational graph of the Message Passing Neural Process. **Input:** the dataset consists of examples (nodes) and a relational structure (edges). Features x are observed at every node, but labels are only available for the context set (the blue nodes labelled C). **Generation:** the encoder $\boldsymbol{h}$ uses message passing operations over the dataset to produce neighbourhood-aware representations of the context set, r_i. The aggregator $\boldsymbol{a}$ combines these into a single representation, r, which parameterises the global latent variable, z. The decoder, $\boldsymbol{g}$, which also uses message-passing operations, is conditioned on a sample from the global latent variable and predicts labels over the target set, $\hat{y}_T$. **Inference:** the predicted labels are added to the target examples, differentiated from the unlabelled targets by the label τ and purple nodes. The dataset is again passed through the encoder and aggregator to produce the global latent variable as conditioned on the joint target and context set, as required in the ELBO objective (Equation 5.9) for training. Note: most message-passing arrows are omitted for clarity.

$|\mathcal{T}| = n$, is then a superset of the context set (which may not necessarily correspond to the entire graph), but contains no labels. When predicting target labels, the decoder g uses information from the k-hop neighbourhoods in a similar manner to the encoder.

5.4.3 Encoder

The encoder h receives as input elements from the context set with initial encodings $\boldsymbol{h}_i = \boldsymbol{x}_i \parallel \boldsymbol{y}_i$ ($\parallel$ denotes concatenation), along with features of nodes from the k-hop neighbourhoods of labelled nodes. Zero-vectors are used to signal the absence of labels, for nodes outside the context set. A representation $\boldsymbol{r}_i$ is produced for each element in the context set by applying T message-passing operations, which are defined in Equation 2.21. Additionally, linear transformations are performed before and after these steps. I now show that, for initial node representations $\boldsymbol{h}_i$, the transformation produced by the encoder:

$$\boldsymbol{r}_i = (L_2 \circ MP^T \circ \text{ReLU} \circ L_1)(\boldsymbol{h}_i) \tag{5.1}$$

is permutation-invariant:

$$\begin{aligned} \forall \text{ permutation } \boldsymbol{\Pi}. (L_2 \circ MP^T \circ \text{ReLU} \circ L_1)(\boldsymbol{X}\boldsymbol{\Pi}, \boldsymbol{\Pi}^\top \boldsymbol{A}\boldsymbol{\Pi}) = \\ \big((L_2 \circ MP \circ \text{ReLU} \circ L_1)(\boldsymbol{X}, \boldsymbol{A})\big)\boldsymbol{\Pi}. \end{aligned} \tag{5.2}$$

Proof: Assume an arbitrary set of features $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ and an adjacency matrix $\boldsymbol{A} \in \{0,1\}^{n \times n}$, where n is the number of nodes in the context set and d is the feature dimensionality. I first prove that each operation within the encoder is permutation-invariant:

1. The linear projections L_1, L_2 are separately applied to each of the node vectors $\boldsymbol{X}_i$. Changing the order of input nodes will thus result in the same order in the output:

$$\begin{aligned} L_i(\boldsymbol{X}\boldsymbol{\Pi}, \boldsymbol{\Pi}^\top \boldsymbol{A}\boldsymbol{\Pi}) &= L_i(\boldsymbol{X}\boldsymbol{\Pi}), \forall i \in \{1,2\} \\ &= (L_i(\boldsymbol{X}_{\Pi_1})\ L_i(\boldsymbol{X}_{\Pi_2})\ \ldots\ L_i(\boldsymbol{X}_{\Pi_n}))^\top \\ &= (L_i(\boldsymbol{X}_1)\ L_i(\boldsymbol{X}_2)\ \ldots\ L_i(\boldsymbol{X}_n))^\top \boldsymbol{\Pi} \\ &= L_i(\boldsymbol{X})\boldsymbol{\Pi} \\ &= L_i(\boldsymbol{X}, \boldsymbol{A})\boldsymbol{\Pi}. \quad \square \end{aligned} \tag{5.3}$$

2. The same result holds for the activation functions, which are applied element-wise:

$$\begin{aligned} \text{ReLU}(\boldsymbol{X}\boldsymbol{\Pi}, \boldsymbol{\Pi}^\top \boldsymbol{A}\boldsymbol{\Pi}) &= \text{ReLU}(\boldsymbol{X}\boldsymbol{\Pi}) \\ &= \text{ReLU}(X_{\Pi_i j}), \forall i, j \\ &= \text{ReLU}(X_{ij})\boldsymbol{\Pi} \\ &= \text{ReLU}(\boldsymbol{X})\boldsymbol{\Pi} \\ &= \text{ReLU}(\boldsymbol{X}, \boldsymbol{A})\boldsymbol{\Pi}. \quad \square \end{aligned} \tag{5.4}$$

3. The message-passing operation is also permutation-invariant, because the transformation $\boldsymbol{A} \to \boldsymbol{P}^\top \boldsymbol{A}\boldsymbol{P}$ preserves the graph structure, with node neighbourhoods undergoing the transformation $\mathcal{N}(i) \triangleq \{j \mid A_{ij} = 1\} \to \mathcal{N}(i)\boldsymbol{\Pi} \triangleq \{\Pi_j \mid A_{\Pi_i \Pi_j} = 1\}$:

$$\begin{aligned} \mathit{MP}(\boldsymbol{X}\boldsymbol{\Pi}, \boldsymbol{\Pi}^\top \boldsymbol{A}\boldsymbol{\Pi}) &= \text{ReLU}\big(\boldsymbol{W}_{\text{skip}}(\boldsymbol{X}\boldsymbol{\Pi})_i + \sum_{j' \in \mathcal{N}(i)\boldsymbol{\Pi}} \boldsymbol{W}_{\text{MP}}(\boldsymbol{X}\boldsymbol{\Pi})_j\big), \text{where } j' = \Pi_j, \\ &= \text{ReLU}\big(\boldsymbol{W}_{\text{skip}}(\boldsymbol{X}_{\Pi_i})\ + \sum_{\Pi_j \in \mathcal{N}(\Pi_i)} \boldsymbol{W}_{\text{MP}}(\boldsymbol{X}_{\Pi_j})\big) \\ &= \text{ReLU}\big((\boldsymbol{W}_{\text{skip}}\boldsymbol{X}_i)\boldsymbol{\Pi}\ + \sum_{j \in \mathcal{N}(i)} (\boldsymbol{W}_{\text{MP}}\boldsymbol{X}_j)\boldsymbol{\Pi}\big) \\ &= \text{ReLU}\big(\boldsymbol{W}_{\text{skip}}\boldsymbol{X}_i\ + \sum_{j \in \mathcal{N}(i)} \boldsymbol{W}_{\text{MP}}\boldsymbol{X}_j\big)\boldsymbol{\Pi} \\ &= \mathit{MP}(\boldsymbol{X}, \boldsymbol{A})\boldsymbol{\Pi}. \quad \square \end{aligned} \tag{5.5}$$

Each type of operation performed within the encoder is thus permutation-invariant. Composing permutation-invariant functions yields a function which has this property itself, so it easily follows that the overall transformation is also permutation-invariant. $\square$

5.4.4 Aggregation

All context node representations $\boldsymbol{r}_i$ are aggregated via a permutation-invariant function $\boldsymbol{a}$, which produces a single vector $\boldsymbol{r} = \boldsymbol{a}(\{\boldsymbol{r}_i\})$, as shown in Figure 5.1. The normal distribution $\boldsymbol{z} \sim \mathcal{N}(\mu_z(\boldsymbol{r}), \mathrm{diag}[\sigma_z(\boldsymbol{r})])$ is associated with the global latent variable $\boldsymbol{z}$, where μ_z and σ_z are linear transformations of $\boldsymbol{r}$.

5.4.5 Decoder

The decoder g receives inputs $\boldsymbol{h}'_i$ which are represented by the concatenations of a sample $\boldsymbol{z}'$ from the distribution described above with each of the target feature vectors: $\boldsymbol{h}'_i = \boldsymbol{x}_i \parallel \boldsymbol{z}'$. This step computes label predictions $\hat{\boldsymbol{y}}_i$ for the nodes in $\mathcal{T}$, in a manner that is similar to the encoding step, producing the output $\boldsymbol{r}'_i$:

$$\boldsymbol{r}'_i = (\mathrm{ReLU} \circ L_2 \circ MP^{\top} \circ \mathrm{ReLU} \circ L_1)(\boldsymbol{h}'_i). \tag{5.6}$$

Following the evaluation convention proposed by Le et al. [107], the target label predictions are computed as follows, where μ_y, σ_y represent linear transformations:

$$\hat{\boldsymbol{y}}_i \sim \mathcal{N}\big(\mathrm{softmax}(\mu_y(\boldsymbol{r}'_i)), \mathrm{diag}[0.1 + 0.9 \times \mathrm{softplus}(\sigma_y(\boldsymbol{r}'_i))]\big). \tag{5.7}$$

5.4.6 Generation and inference

Starting from Equation 2.22, with the function γ corresponding to the neural network $\boldsymbol{g}$ shown in Figure 5.1, and allowing $\boldsymbol{x}_{\mathcal{N}(i)}$ to denote node features from an entire neighbourhood $\mathcal{N}(i)$, the MPNP generative model can be derived as follows:

$$\begin{aligned} p(\boldsymbol{z}, \boldsymbol{y}_{1:n} \mid \boldsymbol{x}_{1:n}, \bigcup_{i=1}^{n} \boldsymbol{x}_{\mathcal{N}(i)}) &= p(\boldsymbol{z}) \prod_{i=1}^{n} p(\boldsymbol{y}_i \mid \boldsymbol{x}_i, \boldsymbol{x}_{\mathcal{N}(i)}, \boldsymbol{z}) \\ &= p(\boldsymbol{z}) \prod_{i=1}^{n} \mathcal{N}\big(\boldsymbol{y}_i \mid \gamma(\boldsymbol{x}_i, \boldsymbol{x}_{\mathcal{N}(i)}, \boldsymbol{z}), \sigma^2\big) \\ &= p(\boldsymbol{z}) \prod_{i=1}^{n} \mathcal{N}\big(\boldsymbol{y}_i \mid F(\boldsymbol{x}_i \| \boldsymbol{z}, \bigodot_{j \in \mathcal{N}(i)}, G(\boldsymbol{x}_i \| \boldsymbol{z}, \boldsymbol{x}_j \| \boldsymbol{z})), \sigma^2\big). \end{aligned} \tag{5.8}$$

In this derivation, line 2 assumes that $p(\boldsymbol{y}_i \mid \boldsymbol{x}_i, \boldsymbol{x}_{\mathcal{N}(i)}, \boldsymbol{z})$ corresponds to a normal distribution, where both mean and variance are functions of $\boldsymbol{x}_i, \boldsymbol{x}_{\mathcal{N}(i)}, \boldsymbol{z}$. Line 3 uses the fact that, in our model, $\gamma = \mathrm{ReLU} \circ L_2 \circ MP^T \circ \mathrm{ReLU} \circ L_1$.

Let us first consider the case for $T = 1$. The function G corresponds to a linear transformation $L_1 = \boldsymbol{W}_{\mathrm{MP}}$ applied to each of the (target node) neighbours' feature vectors

(the concatenated representations $\boldsymbol{x}_j \| \boldsymbol{z}$). This is followed by leveraging the aggregation operation $\bigodot_{j \in \mathcal{N}(i)}$ over the neighbourhood of each target node. Finally, F corresponds to applying a skip-connection (linear transformation) $\boldsymbol{W}_{\text{skip}}$ to each of the target node feature vectors, followed by the ReLU activation of the MP step and ReLU $\circ\, L_2$.

In the case where $T = 2$, the only difference is that the aggregator and linear transformations within the MP step are performed twice. It is important to note that both mean μ and variance σ^2 are output by the same network γ, as each label prediction has its own associated uncertainty.

The decoder function g is obtained by composing several learnable functions (linear projections, MP layers) and non-linearities, thus being trainable with amortised variational inference. The variational posterior $q(\boldsymbol{z}|\boldsymbol{x}_{1:n}, \boldsymbol{y}_{1:n})$ is also parameterised by a permutation-invariant neural network ($\boldsymbol{h}$ in Figure 5.1), as each of the functions in $\boldsymbol{h}$ has this property. Optimisation is achieved using standard methods, via a variational approximation to the ELBO objective, where $D = \boldsymbol{x}_{1:n} \cup \bigcup_{i=1}^{n} \boldsymbol{x}_{\mathcal{N}(i)} \cup \boldsymbol{y}_{1:n}$:

$$\begin{aligned}\log p\big(\boldsymbol{y}_{m+1:n} \mid \boldsymbol{x}_{1:n}, \bigcup_{i=1}^{n} \boldsymbol{x}_{\mathcal{N}(i)}, \boldsymbol{y}_{1:m}\big) \geq \sum_{i=m+1}^{n} \mathbb{E}_{q(\boldsymbol{z}|D)}\big[\log p(\boldsymbol{y}_i \mid \boldsymbol{x}_i, \boldsymbol{x}_{\mathcal{N}(i)}, \boldsymbol{z})\big] \\ - \mathbb{KL}\Big(q(\boldsymbol{z} \mid \boldsymbol{x}_{1:n}, \bigcup_{j=1}^{n} \boldsymbol{x}_{\mathcal{N}(j)}, \boldsymbol{y}_{1:n}) \\ \Big\| q(\boldsymbol{z} \mid \boldsymbol{x}_{1:m}, \bigcup_{j=1}^{m} \boldsymbol{x}_{\mathcal{N}(j)}, \boldsymbol{y}_{1:m})\Big).\end{aligned} \tag{5.9}$$

I now provide a full derivation of the bound stated above, assuming m context nodes and n target nodes (that is, $n-m$ additional targets). We wish to maximise the log-likelihood of the target labels $\boldsymbol{y}_{m+1:n}$, given the target node features $\boldsymbol{x}_{1:n}$, context node features $\boldsymbol{x}_{1:m}$ and labels $\boldsymbol{y}_{1:m}$, and neighbourhoods of context nodes. Features corresponding to an entire neighbourhood are referred to as $\boldsymbol{x}_{\mathcal{N}(i)}$ and $D = \boldsymbol{x}_{1:n} \cup \bigcup_{i=1}^{n} \boldsymbol{x}_{\mathcal{N}(i)} \cup \boldsymbol{y}_{1:n}$.

$$\begin{aligned}&\log p\big(\boldsymbol{y}_{m+1:n} \mid \boldsymbol{x}_{1:n}, \bigcup_{i=1}^{n} \boldsymbol{x}_{\mathcal{N}(i)}, \boldsymbol{y}_{1:m}\big) = \\ &= \log p\big(\boldsymbol{y}_{m+1:n}, \boldsymbol{z} \mid \boldsymbol{x}_{1:n}, \bigcup_{i=1}^{n} \boldsymbol{x}_{\mathcal{N}(i)}, \boldsymbol{y}_{1:m}\big) - \log p\big(\boldsymbol{z} \mid \boldsymbol{x}_{1:n}, \bigcup_{i=1}^{n} \boldsymbol{x}_{\mathcal{N}(i)}, \boldsymbol{y}_{1:n}\big) \\ &= \Big[\log p\big(\boldsymbol{z} \mid \boldsymbol{x}_{1:m}, \bigcup_{i=1}^{m} \boldsymbol{x}_{\mathcal{N}(i)}, \boldsymbol{y}_{1:m}\big) + \sum_{i=m+1}^{n} \log p(\boldsymbol{y}_i \mid \boldsymbol{x}_i, \boldsymbol{x}_{\mathcal{N}(i)}, \boldsymbol{z})\Big] - \\ &\quad \log p\big(\boldsymbol{z} \mid \boldsymbol{x}_{1:n}, \bigcup_{i=1}^{n} \boldsymbol{x}_{\mathcal{N}(i)}, \boldsymbol{y}_{1:n}\big)\end{aligned}$$

$$
\begin{aligned}
&= \log \frac{p\big(\boldsymbol{z} \mid \boldsymbol{x}_{1:m}, \bigcup_{i=1}^{m} \boldsymbol{x}_{\mathcal{N}(i)}, \boldsymbol{y}_{1:m}\big)}{q\big(\boldsymbol{z} \mid \boldsymbol{x}_{1:n}, \bigcup_{i=1}^{n} \boldsymbol{x}_{\mathcal{N}(i)}, \boldsymbol{y}_{1:n}\big)} + \sum_{i=m+1}^{n} \log p(\boldsymbol{y}_i \mid \boldsymbol{x}_i, \boldsymbol{x}_{\mathcal{N}(i)}, \boldsymbol{z}) - \\
&\quad \log \frac{p\big(\boldsymbol{z} \mid \boldsymbol{x}_{1:n}, \bigcup_{i=1}^{n} \boldsymbol{x}_{\mathcal{N}(i)}, \boldsymbol{y}_{1:n}\big)}{q\big(\boldsymbol{z} \mid \boldsymbol{x}_{1:n}, \bigcup_{i=1}^{n} \boldsymbol{x}_{\mathcal{N}(i)}, \boldsymbol{y}_{1:n}\big)} \\
&= \mathbb{E}_{q(\boldsymbol{z}|D)} \left[\sum_{i=1}^{n} \log p(\boldsymbol{y}_i \mid \boldsymbol{x}_i, \boldsymbol{x}_{\mathcal{N}(i)}, \boldsymbol{z}) + \log \frac{p(\boldsymbol{z} \mid \boldsymbol{x}_{1:m}, \bigcup_{j=1}^{m} \boldsymbol{x}_{\mathcal{N}(j)}, \boldsymbol{y}_{1:m})}{q(\boldsymbol{z} \mid \boldsymbol{x}_{1:n}, \bigcup_{j=1}^{n} \boldsymbol{x}_{\mathcal{N}(j)}, \boldsymbol{y}_{1:n})} \right] + \\
&\quad \mathbb{KL}\Big(q(\boldsymbol{z} \mid \boldsymbol{x}_{1:n}, \bigcup_{j=1}^{n} \boldsymbol{x}_{\mathcal{N}(j)}, \boldsymbol{y}_{1:n}) \Big\| p(\boldsymbol{z} \mid \boldsymbol{x}_{1:n}, \bigcup_{j=1}^{n} \boldsymbol{x}_{\mathcal{N}(j)}, \boldsymbol{y}_{1:n}) \Big) \\
&\geq \mathbb{E}_{q(\boldsymbol{z}|D)} \left[\sum_{i=1}^{n} \log p(\boldsymbol{y}_i \mid \boldsymbol{x}_i, \boldsymbol{x}_{\mathcal{N}(i)}, \boldsymbol{z}) + \log \frac{p(\boldsymbol{z} \mid \boldsymbol{x}_{1:m}, \bigcup_{j=1}^{m} \boldsymbol{x}_{\mathcal{N}(j)}, \boldsymbol{y}_{1:m})}{q(\boldsymbol{z} \mid \boldsymbol{x}_{1:n}, \bigcup_{j=1}^{n} \boldsymbol{x}_{\mathcal{N}(j)}, \boldsymbol{y}_{1:n})} \right] \\
&= \sum_{i=m+1}^{n} \mathbb{E}_{q(\boldsymbol{z}|D)} \big[\log p(\boldsymbol{y}_i \mid \boldsymbol{x}_i, \boldsymbol{x}_{\mathcal{N}(i)}, \boldsymbol{z}) \big] - \mathbb{E}_{q(\boldsymbol{z}|D)} \log \frac{q(\boldsymbol{z} \mid \boldsymbol{x}_{1:n}, \bigcup_{j=1}^{n} \boldsymbol{x}_{\mathcal{N}(j)}, \boldsymbol{y}_{1:n})}{p(\boldsymbol{z} \mid \boldsymbol{x}_{1:m}, \bigcup_{j=1}^{m} \boldsymbol{x}_{\mathcal{N}(j)}, \boldsymbol{y}_{1:m})} \\
&= \sum_{i=m+1}^{n} \mathbb{E}_{q(\boldsymbol{z}|D)} \big[\log p(\boldsymbol{y}_i \mid \boldsymbol{x}_i, \boldsymbol{x}_{\mathcal{N}(i)}, \boldsymbol{z}) \big] - \\
&\quad \mathbb{KL}\Big(q(\boldsymbol{z} \mid \boldsymbol{x}_{1:n}, \bigcup_{j=1}^{n} \boldsymbol{x}_{\mathcal{N}(j)}, \boldsymbol{y}_{1:n}) \Big\| q(\boldsymbol{z} \mid \boldsymbol{x}_{1:m}, \bigcup_{j=1}^{m} \boldsymbol{x}_{\mathcal{N}(j)}, \boldsymbol{y}_{1:m}) \Big).
\end{aligned}
$$

In order, the (in)equalities make use of the following results: rewriting the log-likelihood via the posterior distribution, substituting the first term via the generative model, introducing the variational distribution $q(\boldsymbol{z} \mid \boldsymbol{x}_{1:n}, \bigcup_{i=1}^{n} \boldsymbol{x}_{\mathcal{N}(i)}, \boldsymbol{y}_{1:n})$ (here, q corresponds to the composition of the encoder h and aggregator a functions) to approximate the posterior $p(\boldsymbol{z} \mid \boldsymbol{x}_{1:n}, \boldsymbol{y}_{1:n})$, multiplying by $q\big(\boldsymbol{z} \mid \boldsymbol{x}_{1:n}, \bigcup_{i=1}^{n} \boldsymbol{x}_{\mathcal{N}(i)}, \boldsymbol{y}_{1:n}\big)$ and integrating over $\boldsymbol{z}$, the property that $\forall p, q.\ \mathbb{KL}(p \| q) \geq 0$, separating terms, approximating $p(\boldsymbol{z} \mid \boldsymbol{x}_{1:m}, \bigcup_{j=1}^{m} \boldsymbol{x}_{\mathcal{N}(j)}, \boldsymbol{y}_{1:m})$ with $q(\boldsymbol{z} \mid \boldsymbol{x}_{1:m}, \bigcup_{j=1}^{m} \boldsymbol{x}_{\mathcal{N}(j)}, \boldsymbol{y}_{1:m})$ and applying the $\mathbb{KL}$ definition.

5.4.7 Aggregation in challenging settings

The way in which information is represented in the global latent variable $\boldsymbol{z}$ is crucial. At test time, the (context-conditioned) sample from the latent is processed together with the new target points, so it must reflect the behaviour of the new stochastic process, in a way that is relevant to the task. While simply taking the mean over $\boldsymbol{r}_i$ is sufficient for many tasks, we often need to produce a class-aware representation. With this in mind, we adopted the alternative aggregation function used by Garnelo et al. [58] in their few-shot learning experiments:

$$
a'(\{\boldsymbol{r}_i\}) \triangleq \Big\|_{\boldsymbol{c} \in C} a(\mathbb{I}_{\{\boldsymbol{c}\}}(\boldsymbol{y}_i) * \boldsymbol{r}_i). \tag{5.10}
$$

Here, C represents the set of classes in the current context, with their number $|C|$ fixed, as required. This method concatenates ($\|$) all per-class summaries which have been previously aggregated via a. Intuitively, the different classes present in the context set are clearly delimited by this scheme—this is especially helpful in few-shot learning settings, where new classes are seen at test time. We append the '-c' suffix to models that use this scheme, when reporting results.

5.5 Experiments

5.5.1 Baselines and model details

We evaluated MPNPs against a variety of baselines—altogether, these leverage all sources of information which are present in the studied tasks (features, structure & context). This strategy helped us highlight the advantages of the MPNP in each of the settings.

The **label propagation algorithm** (LP) [211] directly uses the context points provided at test time, with nodes being labelled by their neighbours, who in turn label their own neighbours, hence is best suited to tasks that resemble segmentation. Where relevant, we included the mode baseline (**guessing the most common context label**), as this simple approach may significantly outperform the uniform prior ($\frac{1}{|C|}$) on some tasks.

Graph neural networks (GNNs) make use of the training data within an inductive setup, but not the additional context labels provided at test time. We expected them to achieve good performance on tasks where the set of classes is fixed and there is little variation in the generative process across the set of datasets. Crucially, these models are not designed to handle arbitrary labelling tasks and their expected performance is bound by chance ($\mathbb{E}[\text{accuracy}] = \frac{1}{|C|}$). Since predictions do not depend on class labellings, for any given dataset we can construct a set of equivalent datasets by permuting the labels, and over the set of permutations the average performance will be chance. As such, we did not include this baseline on such tasks. In our setup, the GNN consists of GCN layers with skip-connections.

Standard (non-message-passing) **Neural Processes** (NPs) are limited by their inability to leverage relational information between points—this is a straightforward limitation in the setups we consider. We use the same **Message Passing Neural Process** and NP architectures for most tasks.

5.5.1.1 Message passing neural process

The architecture of our model can be summarised as follows:

1. encoder: Linear(h), ReLU, {MP(h), ReLU}$\times T$, Linear(r);
2. global latent variable encoder: Linear(r), [Linear(z), Linear(z)] (mean & variance of $\boldsymbol{z}$);
3. decoder: Linear(h), ReLU, {MP(h), ReLU}$\times T$, Linear(h), ReLU, [Linear(C), Linear(C)] (mean & variance of $\hat{\boldsymbol{y}}$).

Across all experiments, the Adam optimiser is used to maximise the ELBO (that is, to minimise the sum of the negative log-likelihood and KL-divergence in Equation 5.9).

TUD On Enzymes, the MPNP hyperparameters are $h = 64$, $r = 128$, $z = 256$; for the MPNP-c, $h = 64$, $r = 96$, $z = 288$; both have $T = 2$. On DHFR, both MPNP and MPNP-c have $h = 64$, $r = 128$, $z = 256$, $T = 1$. We trained both models for 400 epochs with learning rate $7e \times 10^{-5}$ on DHFR and for 700 epochs with learning rate 1×10^{-4} on Enzymes. For both datasets, we sample context and (additional) target points in the 10%–25% range.

ShapeNet Across all experiments, $h = 64$, $r = 128$, $z = 256$, $T = 2$. The MPNP was trained for 400 epochs on fixed-class and 500 epochs on mixed-class tasks, with 5%–25% context and (additional) target points and a learning rate of 7×10^{-5}.

5.5.1.2 Neural process baseline

The architecture of the NP consists of:

1. encoder: Linear(h), ReLU, Linear(h), ReLU, Linear(r);
2. global latent variable encoder: same as for the MPNP;
3. decoder: Linear(h), ReLU, Linear(h), ReLU, Linear(h), ReLU, [Linear(C), Linear(C)] (mean & variance of $\hat{\boldsymbol{y}}$).

The Adam optimiser is also used here to maximise the ELBO.

TUD On Enzymes, the NP and NP-c hyperparameters are $h = 64, r = 128, z = 512$. On DHFR, both NP and NP-c have $h = 64, r = 64, z = 512$. We trained both models for 400 epochs on DHFR and for 700 epochs on Enzymes, with a learning rate of $4e^{-5}$. For both datasets, we sample 10%–25% context and (additional) target points.

ShapeNet The same hyperparameters were used for all tasks: $h = 64, r = 64, z = 512$. The NP was trained for 400 epochs on fixed-class and 500 epochs on mixed-class tasks, with 5%–25% context and (additional) target points and a learning rate of $4e^{-5}$.

5.5.1.3 Graph neural network baseline

This model consists of 3 GCN [97] layers with skip-connections; each layer computes:

$$\boldsymbol{h}_{t+1} = \text{ReLU}\big(\boldsymbol{W}_{\text{skip}}\boldsymbol{h}_t + \text{GCN}(\boldsymbol{h}_t)\big). \tag{5.11}$$

We use $h = 64$ across all tasks and train the model for 500 epochs, with the Adam optimiser minimising the cross-entropy loss and a learning rate of $1e^{-4}$. The context and target ranges are the same as previously described for each dataset. Once again, I note that this model does not make use of the context labels.

5.5.2 Fixed labelling

When evaluating MPNPs, we first considered tasks where the same set of classes appears in every example and the class labelling is 'fixed'. Inductive GNNs are designed for this setting and provide a useful baseline performance.

5.5.2.1 Biochemical data

I adapted two tasks from the TUD collection [93]: Enzymes and DHFR. Although initially designed for graph classification setups, these datasets also have node-wise labels that allow an alternative node classification scenario.

The Enzymes dataset consists of proteins represented as graphs, where nodes are secondary-structural elements (SSEs—α-helices, β-sheets, β-turns) with associated biochemical features, and edges between elements which are connected. DHFR is a library of small molecule inhibitors against each respective protein target (Dihydrofolate Reductase, Cycloxygenase-2 and the Benzodiazapene Receptor). In the typical graph classification task, molecules are deemed either active or inactive, based on a thresholded half-maximal inhibitory concentration measure that has been determined through *in vitro* biochemical assays. The node classification task I studied here requires the model to predict node labels which correspond to atom types. Node features are *xyz* coordinates of the conformation provided in the datasets and edges represent inter-atomic bonds.

Results Table 5.1 shows the MPNP narrowly outperforming the NP on the Enzymes task. While a much greater margin is noticed for DHFR, in each case the inductive GNN baseline is more successful. This finding suggests that the relational information provided in the Enzymes task is of secondary importance, relative to the features of the SSEs. Furthermore, both datasets most likely have limited variation, given that an inductive GNN model can perform well without any context points. Nevertheless,

	Enzymes			DHFR		
Model	5	10	30	5	10	30
NP	***79.23***	93.43	***95.75***	54.66	55.71	57.38
MPNP	79.09	***94.10***	**95.78**	***88.65***	***89.62***	***90.53***
GNN	**94.23**	**94.23**	94.23	**93.35**	**93.35**	**93.35**
LP	58.93	63.91	76.42	38.48	41.51	53.63

Table 5.1: Node classification on biochemical datasets. Accuracy reported at $\{5, 10, 30\}\%$ context points. **first** / ***second***.

MPNP shows promising ability in using the relational information in DHFR, improving on the NP by a very large margin.

5.5.2.2 Geometric data

The ShapeNet repository [31, 200] is a collection of large-scale 3D shapes, represented as point clouds for our applications. Numerous other encoding methods make fuller use of the geometric information available, but for the purposes of our proof-of-concept study, I only considered only the simplest method. I embedded the points as a nearest-neighbours graph ($\boldsymbol{A}$) and used the (x, y, z) position as node features ($\boldsymbol{X}$). There are 16 object categories, each one with a fixed number of parts (2–6). The labels have consistent meaning across datasets within a category (that is, type of shape). For example, we use the MPNP to model the stochastic process that produces *chairs* with *arms*, *legs*, *seats* and *backs*, which can be consistently labelled with $\{1, 2, 3, 4\}$ across all graphs.

Results Figure 5.2 presents the part-labelling results on ShapeNet. I used the mean-Intersection-over-Union (mIoU) metric, which is commonly used for segmentation tasks. The mIoU is computed as follows: first, find the ratio of overlap (TP) to the union (TP+FP+FN) for each part, then take the average. Here, higher is better, and T/F P/N refers to true/false positives/negatives. I found that the MPNP outperforms the NP at more than 95% confidence on 11 object categories, across the entire context sampling range. Furthermore, MPNPs are the top-performing models over some of the sampling range in 13 out of 15 categories. At 30% sampling rate, label propagation naturally leads, since this is essentially a segmentation task.

5.5.3 Uncertainty modelling

Figure 5.3 shows that the MPNP has superior uncertainty-modelling capabilities. The first 3 plots **visualise the uncertainty estimates** for a *table* sample. The models achieve similar mIoU, but the MPNP is significantly uncertain only at the borders between parts

Dataset	Task	Graphs	Mean-Nodes	Features	Classes
ShapeNet	Bag	76	2749.46	3	2
	Cap	55	2631.53	3	2
	Knife	392	2156.57	3	2
	Laptop	451	2758.13	3	2
	Mug	184	2816.97	3	2
	2-parts	1158	2,557.26	3	2
	Earphone	69	2496.70	3	3
	Guitar	787	2353.91	3	3
	Pistol	283	2654.22	3	3
	Rocket	66	2358.59	3	3
	Skateboard	152	2529.55	3	3
	Table	5271	2722.40	3	3
	3-parts	6628	2,665.34	3	3
	Airplane	2690	2577.92	3	4
	Car	898	2763.81	3	4
	Chair	3758	2705.34	3	4
	Lamp	1547	2198.46	3	4
	4-parts	8893	2,584.53	3	4
	Motorbike	202	2735.65	3	6
TUD	Enzymes	600	32.63	18	3
	DHFR	467	42.23	3	9

Table 5.2: Dataset statistics by tasks.

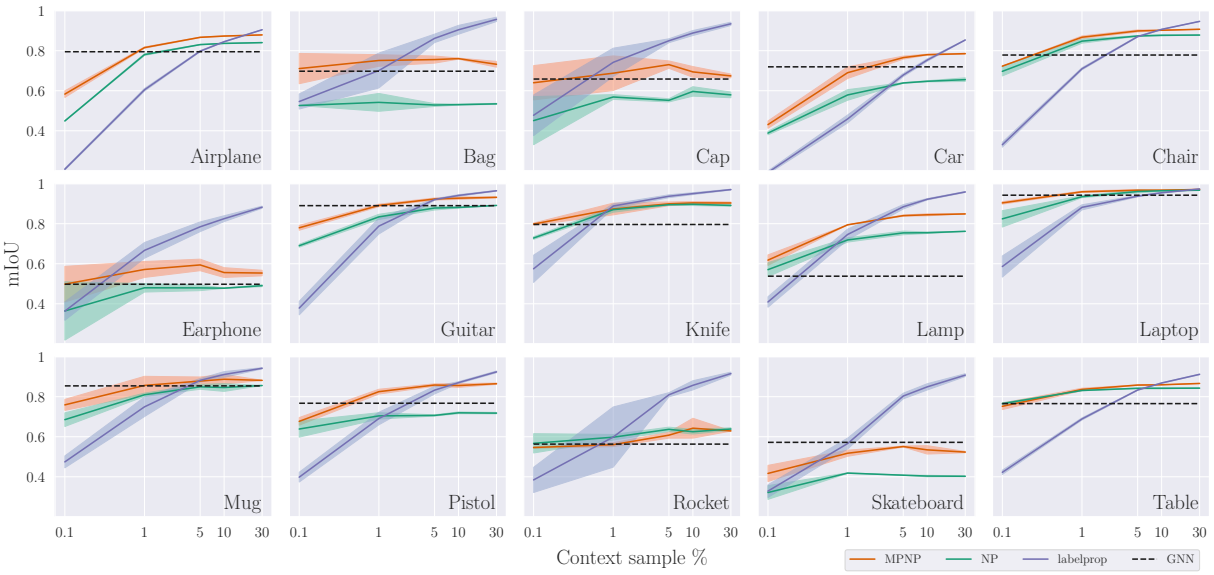

Figure 5.2: Linear-log plots of mIoU over context sample rates with 95% confidence interval shading for the fixed-class ShapeNet task, by category. The GNN is inductive and does not depend on context sampling. Numerical results are given in Appendix A.5.

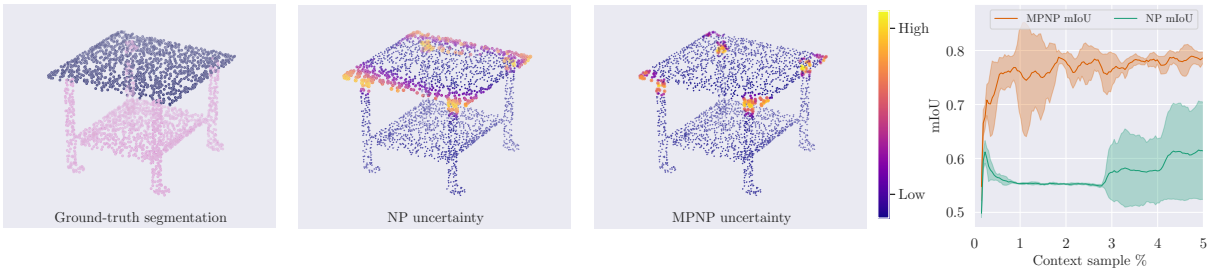

Figure 5.3: Segmentation uncertainty over a test-set example from the ShapeNet fixed-class *table* category and active sampling. (*Left:*) Ground-truth labels are shown for the table-top (purple) and table-leg (pink) parts. (*Centre:*) Uncertainty values are given by the size and colour of the points: higher means larger, yellower points; lower means smaller, bluer points. (*Right:*) Active sampling experiment.

(which corresponds to a physically-relevant uncertainty). In contrast, the NP is more uncertain along the edges of the table top, while in the internal geometry of the table, most of these points are distant from any leg points. On the right, Figure 5.3 shows the results of an **active learning experiment**, carried out in a similar manner to the one described by Garnelo et al. [58]. At each step, all target points are predicted labels and uncertainties. The target point which produced the highest uncertainty estimate is then labelled—that is, added to the context set. The process is then repeated. The rapid increase in mIoU achieved by the MPNP reveals the power of semantically-meaningful uncertainty predictions. Figure 5.4 illustrates additional uncertainty visualisations for other classes in ShapeNet, reinforcing the finding that the estimates produced by MPNPs are semantically relevant.

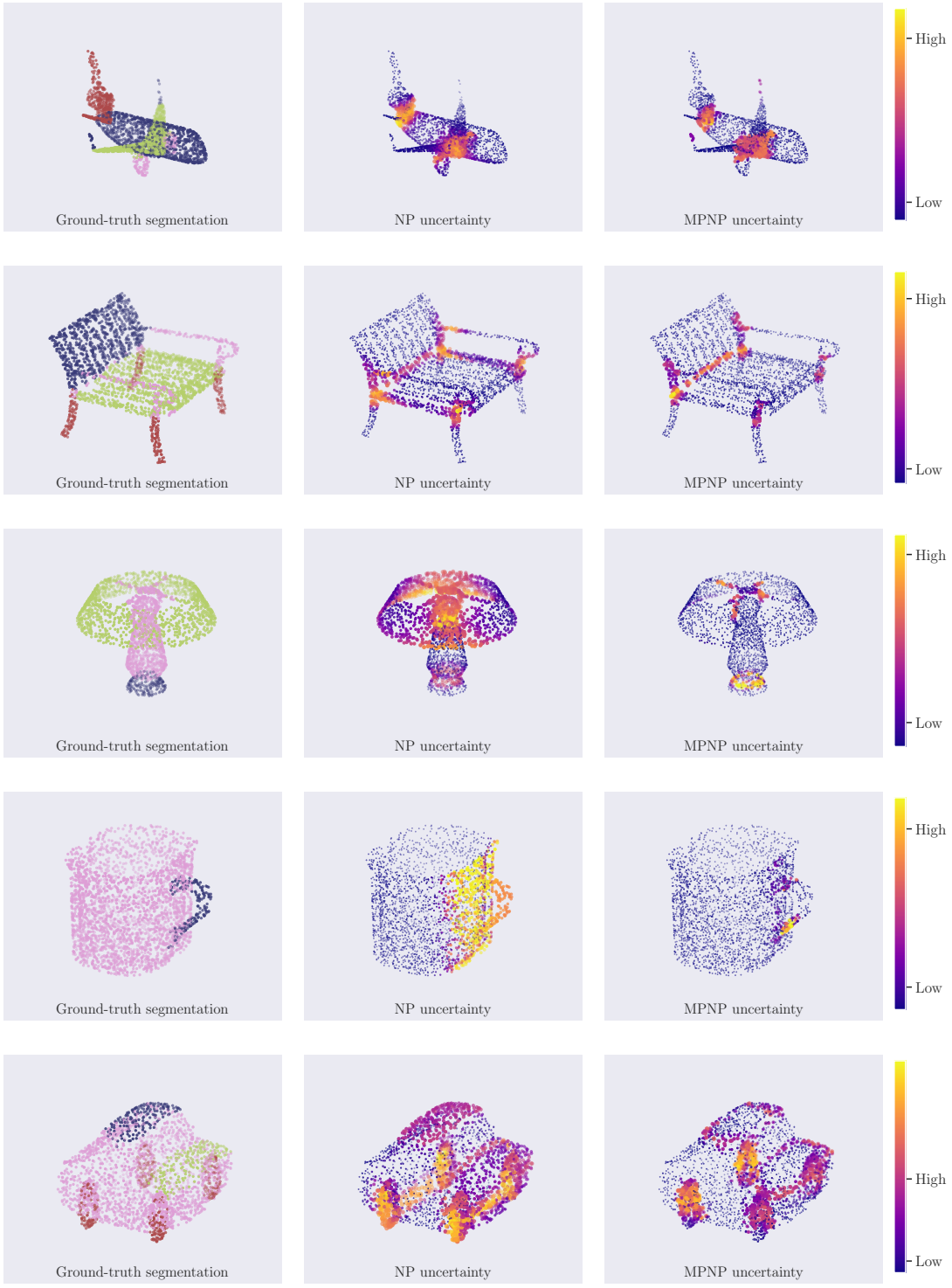

Figure 5.4: Visualisations of the uncertainty estimates on samples from the *airplane*, *chair*, *lamp*, *mug* and *car* ShapeNet categories. In each case, the MPNP can better localise the uncertainty to semantically-relevant regions (namely, borders between parts). Instead, the NP tends to be uncertain on larger and simpler areas—for example, being very uncertain across the entire handle side of the mug. Similar behaviour can be seen for the top of the lamp, the car axles and the edges of the chair-seat. The airplane is a more complex shape, as the borders between wings, fuselage and engines occur in a more focused region; nevertheless, the MPNP is still less uncertain in the airplane-tail.

#	Model	0.1%	1%	5%	10%
2	NP-c	48.06	83.60	88.62	89.17
	MPNP-c	**57.18**	**86.08**	90.81	91.37
	LP	55.55	84.37	**91.90**	**93.93**
	GNN	36.14	36.14	36.14	36.14
3	NP-c	**46.87**	76.66	81.12	81.47
	MPNP-c	45.52	**78.95**	83.80	84.31
	LP	41.12	69.84	**84.40**	**87.76**
	GNN	21.68	21.68	21.68	21.68
4	NP-c	28.48	67.19	72.30	72.88
	MPNP-c	**31.52**	**74.30**	81.38	82.20
	LP	30.29	66.61	**83.61**	**87.91**
	GNN	15.82	15.82	15.82	15.82

Table 5.3: ShapeNet mixed-category, arbitrary-labelling results for 2, 3, and 4-part shapes $(\#)$. We report the mIoU for $\{0.1, 1, 5, 10\}\%$ context points.

5.5.4 Arbitrary labelling

Garnelo et al. [58] applied the Conditional Neural Process in arbitrary-labelling settings. Here, *each dataset* consists of samples that are drawn from a fixed number k of class types, but the total number of types across all datasets is K, with $K \gg k$. As K could be very large and test examples will often include unseen classes, using fixed class indices is infeasible. Instead, arbitrary labellings $(1, ..., k)$ are assigned on a per-dataset basis. Models are then required to discriminate between a novel set of classes each time, as seen from their internal processing perspective.

In the **ShapeNet mixed-category** setup, the MPNP aims to model the stochastic process that produces n-part objects (for example, $n = 4$ for both *chairs*, which have *arms*, *legs*, *seats* and *backs*, and *airplanes*, which are split into *engines*, *bodies*, *tails* and *wings*.) In this setting, labels have consistent meaning only within a given dataset (shape sample), so using a fixed ordering of labels would induce a meaningless learned relationship between, say, *chair-backs* and *airplane-wings* within the model. The class labels are therefore permuted arbitrarily for each example.

Table 5.3 shows the results obtained on the mixed-class part-grouped ShapeNet task. As expected, the GNN baseline struggles to achieve even the simple performance of a random baseline. Label propagation is the most powerful when the context sampling rates are high. In contrast, the MPNP-c and NP-c present relative gains when fewer context points are provided to the respective frameworks. The NP-c performs best at 0.1% on 3-class, disrupting the MPNP-c—however, this may be due to the imbalances across shape categories, as tables represent 80% of all 3-part objects. The MPNP-c and

label propagation generally perform the best, dividing the context sampling range.

5.6 Discussion

I have presented the Message Passing Neural Process, an NP model that leverages the explicit structure between samples from a stochastic process for classification. Our work supplies NPs with the inductive bias necessary to model the relational structure in each dataset, similarly to the ConvCNP model that adds the translation equivariance inductive bias. Therefore, the data points are represented in a context-aware manner, rather than an isolated one. The stronger representations obtained achieve notable performance improvements in few-shot learning and rule-based settings, while uncertainty estimates become more meaningful with respect to the dataset structure.

An important aspect needs to be highlighted, regarding the tasks we chose to evaluate our model on. Recently introduced benchmark suites such as OGB [85] and those proposed by Dwivedi et al. [47] aim to improve the quality of evaluation of graph-based models. However, we found that these collections do not contain tasks that are appropriate for evaluating meta-learning frameworks such as MPNPs—none of the tasks match the problem statement in Section 5.4. To this end, Ben Day designed novel task formulations of the existing ShapeNet [31, 200] and full Cora [17, 123] datasets and entirely novel synthetic tasks based on cellular automata [178, 183, 188]. The results for these tasks can be found in the Appendix (Figure A.1 for cellular automata tasks and Tables A.2 and A.3 for the Cora-Branched transductive and few-shot learning tasks), further supporting the claim that MPNPs build stronger representations from functionally-varied data settings.

Arian Jamasb ran fixed-labelling experiments on a PPI dataset [206]—the task involves predicting which nodes (amino acids) in an amino acid residue graph are involved in an experimentally-determined PPI setting. In this case, the MPNP was able to consistently surpass the R-GCN [161] baseline (which takes into account edge information) across all evaluation metrics (Table A.1 in the Appendix). Solving this task is thought to depend strongly on being able to use relational information, and there is great variation between examples. As expected following the TUD results, the MPNP excels in this setting, with state-of-the-art-competitive results at plausible context rates.

The results presented show that the richer context representations and structural bias of the MPNP are generally beneficial, outperforming the NP on Cora-Branched, PPI, one TUD task and ShapeNet mixed (excluding 3-class with 0.1% context points), while producing semantically-realistic uncertainties, as shown in Figure 5.3 and Figure 5.4. Label propagation is more successful when more labels are available, but MPNP vastly improves on it at low sampling rates, showing powerful capabilities in scarce data

settings. GNNs learn better when the generative process has little functional variation, but perform poorly in the opposite case (mixed-class and few-shot), and are entirely unsuitable in the arbitrary labelling setting. The TUD biochemical datasets are the only fixed-class setting where GNNs do consistently better than MPNPs, though we can attribute this to the lack of functional variation of the generative process in these narrow tasks. As discussed above, results on the PPI task—still within the biochemical domain, but with much greater functional variation—reveal once again that MPNPs are superior to the GNN baseline in these types of settings. On ShapeNet fixed-class tasks, MPNP surpasses the GNN in most cases.

5.7 Summary

In this chapter, I have presented my contribution towards designing and extensively evaluating a new member of the Neural Process family—the Message Passing Neural Process. The main novelty lies in the incorporation of structural inductive biases in the encoder and decoder modules and in the training setup. This constitutes an important addition to the operation of neural processes, since leveraging the connectivity between samples in the context and target sets can be crucial to achieving a stronger functional representation. This is relevant especially for challenging setups, such as ones with restricted availability of the labelled samples. Here, it is important to use all the information that is given to us—including the structural one—towards better distinguishing between sampled functions from the underlying functional distribution.

Our extensive evaluation on various data domains (geometric—ShapeNet, social/citation—Cora-Branched, biochemical—TUD, PPISP, connectionist/rule-based/synthetic—Cellular Automata) showed promising representational power, especially at low sampling rates, where typically powerful algorithms, such as label propagation, underperform. Most importantly for the initial contribution we have proposed in the relational NP space, this study has validated the following hypothesis: adding structural bias to neural process models improves classification performance in settings where stochastic processes also generate structural information between data points.

I consider this type of model to have great potential in domains outside the ones we have studied so far (or had datasets available for). Most interestingly, ML recommender systems could be additionally powered by the graph structure of the associated social network. For example, in a music recommendation context, users of the service and artists (perhaps even individual songs/videos) can be considered different types of nodes in the network graph, with edges of several types denoting kinds of interactions that users have with a certain artist/song: listen, like, add to collection. Another similar

setting can be online shopping, where one can model the network in a similar fashion—nodes are users, products and suppliers; links are purchases, favourites, add to wishlist.

Finally, an important aspect of our study is that the models we have designed and tested leverage general and simple graph processing modules (*linear–MP–linear* in both encoder and decoder networks). Replacing these with more specialised ones, where the message passing scheme is specifically designed for, say, biochemical data, or is represented by more recent and powerful graph convolutional layers or classification models, might boost results even further. In addition, stronger interpretations of the model capabilities may be derived via uncertainty estimate studies, similarly to the ones we conducted to demonstrate the predictive power of Message Passing Neural Processes.

Chapter 6

Conclusion and future directions

This thesis has presented my research efforts carried out during my PhD candidacy. These have been focused on designing methods for informing world representations, which comprise the multimodality and structure present in various types of data scenarios. These two characteristics can be used to better inform ML systems with respect to the downstream task, but also contribute to stronger representations that can be generally leveraged by agents in complex, real-world-like environments. The first contribution [26] looked at *environment understanding from a language-and-vision standpoint*, with the added difficulty of a temporal dimension for the visual stream. The second major effort [16, 25] then focused on building *hierarchical representations of graph-structured data*, while the final piece of work [41] explored graph representations further and addressed *learning functional representations of datasets with relational structure*, which can be used for making predictions in challenging data scenarios. These studies have answered some research questions, which were introduced in Chapter 1—notwithstanding, at least equally many future directions were naturally uncovered.

In **Chapter 3**, I have presented the VideoNavQA benchmark and approaches to learning multimodal representations of indoor environments. The benchmark itself re-imagined the EmbodiedQA task—by removing the navigation requirement from the agent, we allowed the ML system to tackle a much more varied and complex set of question types than in existing instantiations of EQA and other related tasks. The ML systems I designed were based on existing and widely-acknowledged VQA paradigms—FiLM and MAC—which allowed benchmarking the strength of these approaches when the temporal dimension gets added to the visual stream.

The learned representations were tested on question answering downstream tasks, which generally force a certain fusion between the linguistic and visual inputs. Results showed that these models are indeed able to exploit the visual information in order

to produce answers, as a significant gap was present between the performances of multimodal methods and those of question-only baselines. However, the gap to human or perfect[1] performance was still to be bridged at the time of the study, which illustrated the challenge posed by EQA-style tasks for environment understanding, even when components that demand considerable complexity from the system are removed.

Future work in this problem space should focus on interpreting the learned models and designing techniques for tackling the QA tasks present in VideoNavQA, perhaps building upon existing ones, like the framework for video processing proposed by Nicolicioiu et al. [139]. Here, the recurrent space-time graph neural network models the changing world view via a graph of entities and objects with high-level interactions, at both spatial and temporal levels. These types of methods would produce a more precise internal representation of the visual input, in the form of an evolving scene graph across time. Objects would correspond to nodes which get added to the current agent-view graph and removed when they go in and out of the field-of-view, while edges might be learned as denoting proximity-based or spatial relations between objects, such as *on*, *below*, *above*, *behind*, or even correspondence relations, such as *'A chair was seen both in the living room and in the office'*.

Leading researchers have argued [13] in favour of symbolic, concept-focused representations. These should allow an ML system to dedicate its entire capacity to the downstream task itself, rather than being additionally burdened with the task of deciphering the visual input. In the VideoNavQA case—and in visual reasoning tasks, generally—the main bottleneck in solving the task would be matching the visual concepts encountered in the scene to parts of the linguistic conditioning signal. The related research on scene graphs with Boris Knyazev, Eugene Belilovsky and Aaron Courville that I have been involved in illustrated the complexity of this step, within scene graph generation pipelines for visual reasoning. The two resulting studies [99, 100] revealed ways to make these systems more robust to the challenges encountered in the data distributions.

Chapter 4 described two approaches to obtaining hierarchical representations of graph-structured data, based on relational learning with graph neural networks. As real-world environments and contexts can be complex, it is often useful to view and study them at various levels of granularity, a task for which hierarchical representations are highly appropriate. From a supervised learning perspective, these representations are modelled by the downstream task of graph property prediction. Testing such representations may therefore reveal particularities of the given data domain, casting light on the discriminative characteristics of the data across its pre-defined categories.

[1] A human should be able to answer the question, if they viewed the input videos (one frame at a time, in the worst case).

For example, the *'Deep Graph Mapper'* study contains visualisations produced by my collaborator Cristian Bodnar, which illustrate the connection between model prediction errors on input graphs and hierarchical representations.

The first contribution consisted of a CNN-like pipeline for graph classification, where convolutional layers correspond to graph convolutional layers, which perform generalised operations on arbitrarily-structured inputs, instead of grid-like ones, and max-pooling layers are replaced by Top-k graph pooling ones. At the time of publication, no other study had managed to compete with DiffPool, the state-of-the-art pooling method, on the TUD graph classification benchmarks. Essentially for the purpose of the study, we reduced the memory requirement of the classification architecture from $\mathcal{O}(|V|^2)$ to $\mathcal{O}(|V| + |E|)$, where V is the set of nodes and E, the set of edges in the input graph. The authors of Top-k pooling subsequently reported their own graph classification results, which showed the importance of exploring sparse hierarchical approaches to graph classification—required for the eventual scaling to larger, real-world-like inputs.

The second research work combined the expressive capabilities of graph neural networks with the theoretically-proved functionality of a topological data analysis method, Mapper—the resulting pooling layer, MPR, produces topologically-grounded hierarchical representations for graph classification. As mentioned in a previous paragraph, the overarching method of the study, DGM, can be also used for visualising graphs and their node labels (ground-truth or model predictions). Visualisations can be produced at various granularity scales, by appropriately tuning only two parameters. Graph coarsening is achieved via applying a 'lens' ranking function to the nodes—the MPR layer uses the PageRank function, which simply looks at node connectivity and leverages the inductive bias of power-law distributions abundant in social networks. The new nodes are then formed by aggregating all node features within each overlapping interval in the cover $\mathcal{U}$. Despite the 'lens' function being simple and non-differentiable, MPR achieved highly competitive results across nine benchmarks, which deems promising this novel class of pooling methods.

Starting from the approaches that we proposed, an essential future direction in the graph pooling space would combine these two methods. Namely, Top-k pooling and Mapper-based pooling layers could build scalable models that, in turn, produce theoretically-grounded hierarchical representations of graphs using global topological information—something that has not been achieved previously in graph machine learning research, to my knowledge. Moreover, MPR coarsening layers may be designed where the 'lens' function incorporates domain knowledge, which would yield a stronger inductive bias to be leveraged in the node ranking process. Alternatively, one might try to improve the expressive power of the 'lens' function by ranking nodes in more dimensions. In this

setting, it could be useful to leverage node embeddings that were previously extracted and tested on another task.

Chapter 5 introduced a new member of the neural process family, where relational inductive biases were added to accommodate the structure present in functions generated by stochastic processes. Our novel framework, the Message Passing Neural Process, can thus be used for settings akin to node classification and, more generally, for representing a wide and varied functional space for few-shot and multi-task settings. We reported strong performance against different types of baselines, on a wide variety of domains—biochemical, social, geometric, connectionist. MPNPs also yield uncertainty estimates for the class predictions, which can be useful in various settings. One can imagine scoring, for example, how well two molecules interact and bind. A way to achieve this would be by aggregating the uncertainty estimates at node levels from one of the molecules, whilst conditioning the decoding process on a representation of the other molecule—which should be a straightforward extension of the current model.

The study also introduced novel tasks that were devised by Ben, and included several visualisations that highlight the significant improvement in uncertainty modelling achieved by our framework over neural processes. Namely, on ShapeNet, MPNPs properly capture the data semantics, whereas NPs visibly struggle to meaningfully represent the internal geometry of objects. Finally, our experiments revealed the power of adding structure to the encoding and decoding steps present in NP models, at low context sampling rates. Here, the MPNP improves the most over NPs and other powerful baselines such as label propagation, which confirms the gain in representational power obtained from leveraging relational biases.

One can imagine the future members of the neural process family building upon existing ones and creating synergies. For example, world representations benefit from incorporating changes in the inputs across time and space. A spatio-temporal NP framework could thus be designed to account for structural changes—additions and deletions of nodes and links—at various time points. The primary setting where I believe this type of model would be useful is a medical one. In this space, drug design and repurposing is one example of a process that should take into account the evolution of a patient, given the corresponding measurements of drug-gene interactions. Other vital applications where the underlying graphs vary considerably across the distribution and change over time are traffic and weather predictions. In the latter case, one can simply split the forecast map into small regions, or take a more complex approach of constructing a graph out of the various layers of meteorological features present in weather maps.

In a more complex scenario, we can also consider the task of graph completion, starting from a small subgraph (the context) of the final graph (the target). The NP operation mode would undergo some changes, since the process relative to the target set should be similar to the one applied to the context. Generating a graph requires creating additional nodes and links—namely, predicting node features and the existence of edges between (all) pairs of nodes. This would be more challenging, since the encoder needs to behave in the same way as the decoder. A purely generative block might thus be added to the encoder: a suitable choice could be one that is based on normalizing flows, which can learn to transform a sample $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ to a feature vector from a complex distribution.

The overarching aim of this thesis was to **provide building blocks for learning insightful and structured world representations**. The former aspect can be achieved by meaningfully integrating the visual, interactive and linguistic stimuli within an environment, wherein agents reason about the world, make decisions and operate towards completing various real-world tasks. The VideoNavQA work I described in Chapter 3 paves the way to better measuring progress achieved by the research community on QA tasks in indoor environments. The work further proposes vision-and-language reasoning models to tackle the novel task and reports an initial significant edge over language-only baselines, which shows the importance of integrating the two modalities.

The latter aspect of structure is essential to the internal representation of the agent, as its experience in the environment—visual and event-based memory across time—can be modelled into an ever-evolving spatio-temporal graph. Moreover, an agent is likely to gradually build an internal map from exploring its environment. In order to make decisions, this map may need inspection at various levels of granularity (for example, *'Which region should I go to next?'*), which can be achieved using the methods presented in Chapter 4. Finally, agents should be able to act in a variety of settings and be able to generalise to new ones, which is where the Message Passing Neural Process framework from Chapter 5 may prove useful: the agent may, for example, be required to understand outdoor city settings or home environments, which requires making predictions about different sets of object classes. As both structure and multimodality are essential aspects of the world in which we live in, it is crucial for an intelligent agent to combine and learn about these, before reasoning and acting within real-life environments.

Bibliography

[1] Alon Altman. The PageRank Axioms. In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2005.

[2] Jose M. Alvarez, Anelia Angelova, Dhruv Batra, Angel X. Chang, Samyak Datta, Matt Deitke, Ali Farhadi, Aaron Gokaslan, Aleksandra Faust, Jose A. Iglesias-Guitian, Abhishek Kadian, Aniruddha Kembhavi, Vangelis Kokkevis, Vladlen Koltun, Eric Kolve, Stefan Lee, Yongjoon Lee, Eric Li, Antonio Lopez, Oleksandr Maksymets, Jitendra Malik, Roberto Martín-Martín, Roozbeh Mottaghi, Devi Parikh, German Ros, Manolis Savva, Dustin Schwenk, Philipp Slusallek, Julian Straub, Jie Tan, Alexander Toshev, Fei Xia, Erik Wijmans, and Amir Zamir. Embodied AI Workshop, 2020. URL `https://embodied-ai.org/`.

[3] Ankesh Anand, Eugene Belilovsky, Kyle Kastner, Hugo Larochelle, and Aaron Courville. Blindfold Baselines for Embodied QA. *arXiv preprint arXiv:1811.05013*, 2018.

[4] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On Evaluation of Embodied Navigation Agents. *arXiv preprint arXiv:1807.06757*, 2018.

[5] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-Language Navigation: Interpreting Visually-Grounded Navigation Instructions in Real Environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[6] Anonymous. Graph U-Net. In *Submitted to the Seventh International Conference on Learning Representations (ICLR)*, 2018. Under review.

[7] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. VQA: Visual question answering. In

Proceedings of the IEEE International Conference on Computer Vision, pages 2425–2433, 2015.

[8] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant Risk Minimization. *arXiv preprint arXiv:1907.02893*, 2019.

[9] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

[10] Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron Courville. Systematic Generalization: What Is Required and Can It Be Learned? *arXiv preprint arXiv:1811.12889*, 2018.

[11] Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya Nori, and Antonio Criminisi. Measuring Neural Net Robustness with Constraints. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29, pages 2613–2621. Curran Associates, Inc., 2016.

[12] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[13] Yoshua Bengio. From System 1 Deep Learning to System 2 Deep Learning (Posner lecture at NeurIPS), 2019. URL `http://www.iro.umontreal.ca/~bengioy/NeurIPS-11dec2019.pdf`.

[14] Yoshua Bengio, Tristan Deleu, Nasim Rahaman, Nan Rosemary Ke, Sebastien Lachapelle, Olexa Bilaniuk, Anirudh Goyal, and Christopher Pal. A Meta-Transfer Objective for Learning to Disentangle Causal Mechanisms. In *International Conference on Learning Representations*, 2020.

[15] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Mincut Pooling in Graph Neural Networks. *arXiv preprint arXiv:1907.00481*, 2019.

[16] Cristian Bodnar, Cătălina Cangea, and Pietro Liò. Deep Graph Mapper: Seeing Graphs through the Neural Lens. *NeurIPS Topological Data Analysis and Beyond Workshop*, 2020.

[17] Aleksandar Bojchevski and Stephan Günnemann. Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking. In *International Conference on Learning Representations*, pages 1–13, 2018.

[18] Paolo Boldi, Massimo Santini, and Sebastiano Vigna. PageRank as a Function of the Damping Factor. In *Proceedings of the 14th International Conference on World Wide Web*, pages 557–566, 2005.

[19] Simon Brodeur, Ethan Perez, Ankesh Anand, Florian Golemo, Luca Celotti, Florian Strub, Jean Rouat, Hugo Larochelle, and Aaron Courville. HoME: A Household Multimodal Environment. *arXiv preprint arXiv:1711.11017*, 2017.

[20] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[21] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[22] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[23] Daphna Buchsbaum, Thomas L. Griffiths, Dillon Plunkett, Alison Gopnik, and Dare Baldwin. Inferring action structure and causal relationships in continuous sequences of human action. *Cognitive Psychology*, 76:30–77, 2015. ISSN 0010-0285. doi: https://doi.org/10.1016/j.cogpsych.2014.10.001.

[24] Remi Cadene, Hedi Ben-Younes, Nicolas Thome, and Matthieu Cord. MUREL: Multimodal Relational Reasoning for Visual Question Answering. In *IEEE Conference on Computer Vision and Pattern Recognition CVPR*, 2019.

[25] Cătălina Cangea, Petar Veličković, Nikola Jovanović, Thomas Kipf, and Pietro Liò. Towards Sparse Hierarchical Graph Classifiers. *NeurIPS Relational Representation Learning Workshop*, 2018.

[26] Cătălina Cangea, Eugene Belilovsky, Pietro Liò, and Aaron Courville. VideoNavQA: Bridging the Gap between Visual and Embodied Question Answering. In *British Machine Vision Conference (BMVC)*, 2019.

[27] Cătălina Cangea, Abhishek Das, Drew Hudson, Jacob Krantz, Stefan Lee, Jiayuan Mao, Florian Strub, Alane Suhr, and Erik Wijmans. Visually Grounded Interaction and Language Workshop, 2021. URL `https://vigilworkshop.github.io/`.

[28] Kris Cao, Angeliki Lazaridou, Marc Lanctot, Joel Z Leibo, Karl Tuyls, and Stephen Clark. Emergent Communication through Negotiation. In *International Conference on Learning Representations*, 2018.

[29] Andrew Carr and David Wingate. Graph Neural Processes: Towards Bayesian Graph Neural Networks. *arXiv e-prints*, art. arXiv:1902.10042, February 2019.

[30] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. Everybody dance now. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5933–5942, 2019.

[31] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. *arXiv e-prints*, art. arXiv:1512.03012, December 2015.

[32] Frédéric Chazal and Bertrand Michel. An introduction to Topological Data Analysis: Fundamental and practical aspects for data scientists. *arXiv preprint arXiv:1710.04019*, 2017.

[33] Howard Chen, Alane Suhr, Dipendra Misra, Noah Snavely, and Yoav Artzi. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12538–12547, 2019.

[34] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[35] Wei Chu, Vikas Sindhwani, Zoubin Ghahramani, and S Sathiya Keerthi. Relational Learning with Gaussian Processes. In *Advances in Neural Information Processing Systems*, pages 289–296, 2007.

[36] Rodolfo Corona, Daniel Fried, Coline Devin, Dan Klein, and Trevor Darrell. Modularity Improves Out-of-Domain Instruction Following. *arXiv preprint arXiv:2010.12764*, 2020.

[37] Hanjun Dai, Bo Dai, and Le Song. Discriminative embeddings of latent variable models for structured data. In *International Conference on Machine Learning*, pages 2702–2711, 2016.

[38] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied Question Answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[39] Abhishek Das, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Neural Modular Control for Embodied Question Answering. *arXiv preprint arXiv:1810.11181*, 2018.

[40] Abhishek Das, Federico Carnevale, Hamza Merzic, Laura Rimell, Rosalia Schneider, Josh Abramson, Alden Hung, Arun Ahuja, Stephen Clark, Greg Wayne, and Felix Hill. Probing Emergent Semantics in Predictive Agents via Question Answering. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 2376–2391. PMLR, 13–18 Jul 2020.

[41] Ben Day, Cătălina Cangea, Arian R Jamasb, and Pietro Liò. Message Passing Neural Processes. *arXiv preprint arXiv:2009.13895*, 2020.

[42] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016.

[43] Nima Dehmamy, Albert-Laszlo Barabasi, and Rose Yu. Understanding the Representation Power of Graph Neural Networks in Learning Graph Topology. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[44] Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, Luca Weihs, Mark Yatskar, and Ali Farhadi. RoboTHOR: An Open Simulation-to-Real Embodied AI Platform. In *CVPR*, 2020.

[45] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Weighted Graph Cuts without Eigenvectors A Multilevel Approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11), 2007.

[46] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional Networks on Graphs for Learning Molecular Fingerprints. In *Advances in Neural Information Processing Systems*, pages 2224–2232, 2015.

[47] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking Graph Neural Networks. *arXiv preprint arXiv:2003.00982*, 2020.

[48] Paul Erdos and Alfréd Rényi. On the evolution of random graphs. *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*, 5(1):17–60, 1960.

[49] Richard Evans and Edward Grefenstette. Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research*, 61:1–64, 2018.

[50] Katrina Evtimova, Andrew Drozdov, Douwe Kiela, and Kyunghyun Cho. Emergent Communication in a Multi-Modal, Multi-Step Referential Game. In *International Conference on Learning Representations*, 2018.

[51] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. SplineCNN: Fast geometric deep learning with continuous B-spline kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 869–877, 2018.

[52] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[53] Katerina Fragkiadaki. Language Grounding to Vision and Control, Fall 2017 CMU 10-808, 2017. URL `https://katefvision.github.io/LanguageGrounding/`.

[54] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. Multimodal Compact Bilinear Pooling for Visual Question Answering and Visual Grounding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 457–468, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1044.

[55] Chuang Gan, Yiwei Zhang, Jiajun Wu, Boqing Gong, and Joshua B Tenenbaum. Look, listen, and act: Towards audio-visual embodied navigation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9701–9707. IEEE, 2020.

[56] Andrew Gao, Ruohan Owens, Dinesh Jayaraman, Yuke Zhu, Jiajun Wu, , and Kristen Grauman. Embodied Multimodal Workshop, 2021. URL `https://eml-workshop.github.io/`.

[57] Hongyang Gao and Shuiwang Ji. Graph U-Nets. In *International Conference on Machine Learning*, pages 2083–2092, 2019.

[58] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional Neural Processes. In *International Conference on Machine Learning*, pages 1704–1713, 2018.

[59] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural Processes. *arXiv preprint arXiv:1807.01622*, 2018.

[60] Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, and Aviv Tamar. Bayesian Reinforcement Learning: A Survey. *Foundations and Trends in Machine Learning*, 8 (5–6):359–483, November 2015. ISSN 1935-8237. doi: 10.1561/2200000049.

[61] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural Message Passing for Quantum Chemistry. 2017.

[62] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.

[63] Vik Goel, Jameson Weng, and Pascal Poupart. Unsupervised Video Object Segmentation for Deep Reinforcement Learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 5688–5699, Red Hook, NY, USA, 2018. Curran Associates Inc.

[64] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, volume 27, pages 2672–2680. Curran Associates, Inc., 2014.

[65] Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. IQA: Visual question answering in interactive environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4089–4098, 2018.

[66] Jonathan Gordon, Wessel P. Bruinsma, Andrew Y. K. Foong, James Requeima, Yann Dubois, and Richard E. Turner. Convolutional Conditional Neural Processes. *arXiv e-prints*, art. arXiv:1910.13556, October 2019.

[67] Gaurav Goswami, Nalini Ratha, Akshay Agarwal, Richa Singh, and Mayank Vatsa. Unravelling robustness of deep learning based face recognition against adversarial attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[68] Jonathan Gratch, Jeff Rickel, Elisabeth André, Justine Cassell, Eric Petajan, and Norman Badler. Creating interactive virtual humans: Some assembly required. *IEEE Intelligent systems*, 17(4):54–63, 2002.

[69] Raia Hadsell, Dushyant Rao, Andrei A Rusu, and Razvan Pascanu. Embracing Change: Continual Learning in Deep Neural Networks. *Trends in Cognitive Sciences*, 2020.

[70] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring Network Structure, Dynamics, and Function using NetworkX. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[71] Mustafa Hajij, Paul Rosen, and Bei Wang. Mapper on Graphs for Network Visualization, 2018.

[72] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[73] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

[74] Weituo Hao, Chunyuan Li, Xiujun Li, Lawrence Carin, and Jianfeng Gao. Towards Learning a Generic Agent for Vision-and-Language Navigation via Pre-Training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[75] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[76] Dan Hendrycks and Thomas Dietterich. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. In *International Conference on Learning Representations*, 2019.

[77] Karl Moritz Hermann, Mateusz Malinowski, Piotr Mirowski, Andras Banki-Horvath, Keith Anderson, and Raia Hadsell. Learning to follow directions in Street View. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11773–11781, 2020.

[78] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining Improvements in Deep Reinforcement Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[79] Matteo Hessel, Hubert Soyer, Lasse Espeholt, Wojciech Czarnecki, Simon Schmitt, and Hado van Hasselt. Multi-task Deep Reinforcement Learning with PopArt.

In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3796–3803, 2019.

[80] Felix Hill, Andrew Lampinen, Rosalia Schneider, Stephen Clark, Matthew Botvinick, James L. McClelland, and Adam Santoro. Environmental drivers of systematicity and generalization in a situated agent. In *International Conference on Learning Representations*, 2020.

[81] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735.

[82] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3588–3597, 2018.

[83] Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 804–813, 2017.

[84] Ronghang Hu, Daniel Fried, Anna Rohrbach, Dan Klein, Trevor Darrell, and Kate Saenko. Are You Looking? Grounding to Multiple Modalities in Vision-and-Language Navigation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019*, pages 6551–6557. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1655.

[85] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open Graph Benchmark: Datasets for Machine Learning on Graphs. *arXiv preprint arXiv:2005.00687*, 2020.

[86] Jingjia Huang, Zhangheng Li, Nannan Li, Shan Liu, and Ge Li. AttPool: Towards Hierarchical Feature Representation in Graph Convolutional Networks via Attention Mechanism. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6480–6489, 2019.

[87] Drew A Hudson and Christopher D Manning. GQA: a new dataset for compositional question answering over real-world images. *arXiv preprint arXiv:1902.09506*, 2019.

[88] Drew Arad Hudson and Christopher D. Manning. Compositional Attention Networks for Machine Reasoning. In *International Conference on Learning Representations*, 2018.

[89] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 448–456, Lille, France, 07–09 Jul 2015.

[90] Max Jaderberg, Wojciech M. Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castañeda, Charles Beattie, Neil C. Rabinowitz, Ari S. Morcos, Avraham Ruderman, Nicolas Sonnerat, Tim Green, Louise Deason, Joel Z. Leibo, David Silver, Demis Hassabis, Koray Kavukcuoglu, and Thore Graepel. Human-level performance in 3D multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019. ISSN 0036-8075. doi: 10.1126/science.aau6249.

[91] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910, 2017.

[92] Nan Rosemary Ke, Olexa Bilaniuk, Anirudh Goyal, Stefan Bauer, Hugo Larochelle, Bernhard Schölkopf, Michael C Mozer, Chris Pal, and Yoshua Bengio. Learning neural causal models from unknown interventions. *arXiv preprint arXiv:1910.01075*, 2019.

[93] Kristian Kersting, Nils M. Kriege, Christopher Morris, Petra Mutzel, and Marion Neumann. Benchmark data sets for graph kernels, 2016. URL `http://graphkernels.cs.tu-dortmund.de`.

[94] Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive Neural Processes. *arXiv e-prints*, art. arXiv:1901.05761, January 2019.

[95] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[96] Thomas Kipf, Elise van der Pol, and Max Welling. Contrastive Learning of Structured World Models. In *International Conference on Learning Representations*, 2020.

[97] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. 2016.

[98] Stephen C. Kleene. Representations of events in nerve nets and finite automata. *Automata Studies* [*Annals of Math. Studies 34*], 1956.

[99] Boris Knyazev, Harm de Vries, Cătălina Cangea, Graham W. Taylor, Aaron Courville, and Eugene Belilovsky. Generative Graph Perturbations for Scene Graph Prediction. *ICML Workshop on Object-Oriented Learning (OOL): Perception, Representation, and Reasoning*, 2020.

[100] Boris Knyazev, Harm de Vries, Cătălina Cangea, Graham W Taylor, Aaron Courville, and Eugene Belilovsky. Graph Density-Aware Losses for Novel Compositions in Scene Graph Generation. In *British Machine Vision Conference (BMVC)*, 2020.

[101] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv preprint arXiv:1712.05474*, 2017.

[102] Carly Kontra, Susan Goldin-Meadow, and Sian L Beilock. Embodied learning across the life span. *Topics in Cognitive Science*, 4(4):731–739, 2012.

[103] Vivek Kothari, Catherine Tong, and Nicholas Lane. The Surprising Behavior Of Graph Neural Networks, 2020. URL `https://openreview.net/forum?id=BkgOM1rKvr`.

[104] Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the Nav-Graph: Vision-and-Language Navigation in Continuous Environments. In *Computer Vision – ECCV 2020*, pages 104–120, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58604-1.

[105] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[106] David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Rémi Le Priol, Dinghuai Zhang, and Aaron Courville. Out-of-Distribution Generalization via Risk Extrapolation ({RE}x), 2021.

[107] Tuan Anh Le, Hyunjik Kim, Marta Garnelo, Dan Rosenbaum, Jonathan Schwarz, and Yee Whye Teh. Empirical Evaluation of Neural Process Objectives. In *NeurIPS workshop on Bayesian Deep Learning*, 2018.

[108] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. ISSN 00189219. doi: 10.1109/5.726791.

[109] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[110] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, R. Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten Digit Recognition with a Back-Propagation Network. In *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1990.

[111] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-Attention Graph Pooling. In *International Conference on Machine Learning*, pages 3734–3743, 2019.

[112] Jie Lei, Licheng Yu, Mohit Bansal, and Tamara L Berg. TVQA: Localized, Compositional Video Question Answering. In *EMNLP*, 2018.

[113] Ang Li, Huiyi Hu, Piotr Mirowski, and Mehrdad Farajtabar. Cross-view policy learning for street navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8100–8109, 2019.

[114] Ruiyu Li, Makarand Tapaswi, Renjie Liao, Jiaya Jia, Raquel Urtasun, and Sanja Fidler. Situation Recognition with Graph Neural Networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4173–4182, 2017.

[115] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[116] Y. C. Liu, J. Tian, C. Y. Ma, N. Glaser, C. W. Kuo, and Z. Kira. Who2com: Collaborative Perception via Learnable Handshake Communication. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6876–6883, 2020. doi: 10.1109/ICRA40945.2020.9197364.

[117] Christos Louizos, Xiahan Shi, Klamer Schutte, and Max Welling. The Functional Neural Process. In *Advances in Neural Information Processing Systems 32*, pages 8746–8757. Curran Associates, Inc., 2019.

[118] Enxhell Luzhnica, Ben Day, and Pietro Lio. Clique pooling for graph classification. *arXiv preprint arXiv:1904.00374*, 2019.

[119] Chih-Yao Ma, Zuxuan Wu, Ghassan AlRegib, Caiming Xiong, and Zsolt Kira. The regretful agent: Heuristic-aided navigation through progress estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6732–6740, 2019.

[120] Yao Ma, Suhang Wang, Charu C Aggarwal, and Jiliang Tang. Graph Convolutional Networks with EigenPooling. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 723–731, 2019.

[121] Mateusz Malinowski and Mario Fritz. A Multi-World Approach to Question Answering about Real-World Scenes based on Uncertain Input. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

[122] Mateusz Malinowski, Carl Doersch, Adam Santoro, and Peter Battaglia. Learning Visual Question Answering by Bootstrapping Hard Attention. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–20, 2018.

[123] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the Construction of Internet Portals with Machine Learning. *Information Retrieval*, 3(2):127–163, 2000. ISSN 13864564. doi: 10.1023/A:1009953814988.

[124] Péter Mernyei and Cătălina Cangea. Wiki-CS: A Wikipedia-based Benchmark for Graph Neural Networks. *ICML Graph Representation Learning and Beyond Workshop*, 2020.

[125] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016.

[126] Piotr Mirowski, Matt Grimes, Mateusz Malinowski, Karl Moritz Hermann, Keith Anderson, Denis Teplyashin, Karen Simonyan, koray kavukcuoglu, Andrew Zisserman, and Raia Hadsell. Learning to Navigate in Cities Without a Map. In *Advances in Neural Information Processing Systems*, volume 31, pages 2419–2430. Curran Associates, Inc., 2018.

[127] Piotr Mirowski, Andras Banki-Horvath, Keith Anderson, Denis Teplyashin, Karl Moritz Hermann, Mateusz Malinowski, Matthew Koichi Grimes, Karen Simonyan, Koray Kavukcuoglu, Andrew Zisserman, et al. The StreetLearn environment and dataset. *arXiv preprint arXiv:1903.01292*, 2019.

[128] Dmytro Mishkin, Alexey Dosovitskiy, and Vladlen Koltun. Benchmarking Classic and Learned Navigation in Complex 3D Environments. *arXiv preprint arXiv:1901.10915*, 2019.

[129] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems*, pages 2265–2273, 2013.

[130] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[131] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[132] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937. PMLR, 2016.

[133] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model CNNs. In *Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, page 3, 2017.

[134] Damian Mrowca, Chengxu Zhuang, Elias Wang, Nick Haber, Li Fei-Fei, Joshua B Tenenbaum, and Daniel LK Yamins. Flexible Neural Representation for Physics Prediction. *arXiv preprint arXiv:1806.08047*, 2018.

[135] Jonghwan Mun, Paul Hongsuck Seo, Ilchae Jung, and Bohyung Han. MarioQA: Answering Questions by Watching Gameplay Videos. In *ICCV*, 2017.

[136] Medhini Narasimhan, Svetlana Lazebnik, and Alexander Schwing. Out of the box: Reasoning with graph convolution nets for factual visual question answering. In *Advances in Neural Information Processing Systems*, pages 2654–2665, 2018.

[137] Yin Cheng Ng, Nicolò Colombo, and Ricardo Silva. Bayesian Semi-Supervised Learning with Graph Gaussian Processes. In *Advances in Neural Information Processing Systems*, pages 1683–1694, 2018.

[138] Dat Tien Nguyen, Shikhar Sharma, Hannes Schulz, and Layla El Asri. From FiLM to Video: Multi-turn Question Answering with Multi-modal Context. *arXiv preprint arXiv:1812.07023*, 2018.

[139] Andrei Nicolicioiu, Iulia Duta, and Marius Leordeanu. Recurrent Space-time Graph Neural Networks. In *Advances in Neural Information Processing Systems*, volume 32, pages 12838–12850. Curran Associates, Inc., 2019.

[140] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International Conference on Machine Learning*, pages 2014–2023, 2016.

[141] Will Norcliffe-Brown, Stathis Vafeias, and Sarah Parisot. Learning conditioned graph structures for interpretable visual question answering. In *Advances in Neural Information Processing Systems*, pages 8334–8343, 2018.

[142] Y. Ofran and B. Rost. ISIS: Interaction Sites Identified from Sequence. *Bioinformatics*, 23(2):e13–e16, January 2007. doi: 10.1093/bioinformatics/btl303.

[143] Felix L. Opolka and Pietro Liò. Graph Convolutional Gaussian Processes For Link Prediction. *arXiv e-prints*, art. arXiv:2002.04337, February 2020.

[144] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford InfoLab, 1999.

[145] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[146] Rolf Pfeifer and Fumiya Iida. Embodied artificial intelligence: Trends and challenges. In *Embodied Artificial Intelligence*, pages 1–26. Springer, 2004.

[147] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. VirtualHome: Simulating Household Activities via Programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8494–8502, 2018.

[148] Siyuan Qi, Wenguan Wang, Baoxiong Jia, Jianbing Shen, and Song-Chun Zhu. Learning Human-Object Interactions by Graph Parsing Neural Networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 401–417, 2018.

[149] Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel. REVERIE: Remote Embodied Visual Referring Expression in Real Indoor Environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9982–9991, 2020.

[150] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-Learning with Implicit Gradients. In *Advances in Neural Information Processing Systems*, volume 32, pages 113–124. Curran Associates, Inc., 2019.

[151] Ekagra Ranjan, Soumya Sanyal, and Partha Pratim Talukdar. ASAP: Adaptive Structure Aware Pooling for Learning Hierarchical Graph Representations. *arXiv preprint arXiv:1911.07979*, 2019.

[152] Dushyant Rao, Francesco Visin, Andrei A Rusu, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Continual unsupervised representation learning. *arXiv preprint arXiv:1910.14481*, 2019.

[153] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2017.

[154] Mengye Ren, Ryan Kiros, and Richard Zemel. Exploring models and data for image question answering. In *Advances in Neural Information Processing Systems*, pages 2953–2961, 2015.

[155] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[156] James Requeima, Jonathan Gordon, John Bronskill, Sebastian Nowozin, and Richard E Turner. Fast and Flexible Multi-Task Classification using Conditional Neural Adaptive Processes. In *Advances in Neural Information Processing Systems*, pages 7957–7968, 2019.

[157] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.

[158] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. 2015.

[159] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A Platform for Embodied AI Research. *arXiv preprint arXiv:1904.01201*, 2019.

[160] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[161] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling Relational Data with Graph Convolutional Networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018.

[162] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-Lehman Graph Kernels. *Journal of Machine Learning Research*, 12(77):2539–2561, 2011.

[163] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[164] Martin Simonovsky and Nikos Komodakis. Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs. In *Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[165] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[166] Gautam Singh, Jaesik Yoon, Youngsung Son, and Sungjin Ahn. Sequential Neural Processes. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 10254–10264. Curran Associates, Inc., 2019.

[167] Gurjeet Singh, Facundo Memoli, and Gunnar Carlsson. Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition, 2007.

[168] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic Scene Completion from a Single Depth Image. *Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[169] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[170] Florian Strub, Mathieu Seurin, Ethan Perez, Harm De Vries, Jérémie Mary, Philippe Preux, Aaron Courville, and Olivier Pietquin. Visual Reasoning with Multi-hop Feature Modulation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–800, 2018.

[171] Raphael Suter, Djordje Miladinovic, Bernhard Schölkopf, and Stefan Bauer. Robustly disentangled causal mechanisms: Validating deep representations for interventional robustness. In *International Conference on Machine Learning*, pages 6056–6065. PMLR, 2019.

[172] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems*, volume 27, pages 3104–3112. Curran Associates, Inc., 2014.

[173] Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. MovieQA: Understanding Stories in Movies through Question-Answering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[174] Damien Teney, Lingqiao Liu, and Anton van den Hengel. Graph-structured representations for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2017.

[175] Jesse Thomason, Michael Murray, Maya Cakmak, and Luke Zettlemoyer. Vision-and-dialog navigation. In *Conference on Robot Learning*, pages 394–406. PMLR, 2020.

[176] Jesse Thomason, Aishwarya Padmakumar, Jivko Sinapov, Nick Walker, Yuqian Jiang, Harel Yedidsion, Justin Hart, Peter Stone, and Raymond Mooney. Jointly improving parsing and perception for natural language commands through human-robot dialog. *Journal of Artificial Intelligence Research*, 67:327–374, 2020.

[177] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3D convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4489–4497, 2015.

[178] Alan Mathison Turing. The Chemical Basis of Morphogenesis. *Bulletin of Mathematical Biology*, 52(1-2):153–197, 1990.

[179] Hado Van Hasselt, Arthur Guez, and David Silver. Deep Reinforcement Learning with Double Q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[180] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc., 2017.

[181] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018.

[182] Paul Vogt. Language evolution and robotics: issues on symbol grounding and language acquisition. In *Artificial Cognition Systems*, pages 176–209. IGI Global, 2007.

[183] John Von Neumann et al. The General and Logical Theory of Automata. *1951*, pages 1–41, 1951.

[184] Ian Walker and Ben Glocker. Graph Convolutional Gaussian Processes. In *International Conference on Machine Learning*, pages 6495–6504, 2019.

[185] Xin Wang, Ronghang Hu, Drew Hudson, Tsu-Jui Fu, Marcus Rohrbach, and Daniel Fried. Advances in Language and Vision Research Workshop, 2021. URL `https://alvr-workshop.github.io/`.

[186] Erik Wijmans, Samyak Datta, Oleksandr Maksymets, Abhishek Das, Georgia Gkioxari, Stefan Lee, Irfan Essa, Devi Parikh, and Dhruv Batra. Embodied Question Answering in Photorealistic Environments with Point Cloud Perception. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[187] Christopher KI Williams and Carl Edward Rasmussen. Gaussian Processes for Regression. In *Advances in Neural Information Processing Systems*, pages 514–520, 1996.

[188] Stephen Wolfram. Universality and Complexity in Cellular Automata. *Physica D: Nonlinear Phenomena*, 10(1-2):1–35, 1984.

[189] Min Wu and Marta Kwiatkowska. Robustness Guarantees for Deep Neural Networks on Videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[190] Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. Building generalizable agents with a realistic and rich 3D environment. *arXiv preprint arXiv:1801.02209*, 2018.

[191] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson Env: Real-World Perception for Embodied Agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9068–9079, 2018.

[192] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. SAPIEN: A SimulAted Part-based Interactive ENvironment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11097–11107, 2020.

[193] Jin Xu, Jean-Francois Ton, Hyunjik Kim, Adam Kosiorek, and Yee Whye Teh. MetaFun: Meta-Learning with Iterative Functional Updates. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 10617–10627. PMLR, 13–18 Jul 2020.

[194] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[195] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation Learning on Graphs with Jumping Knowledge Networks. *arXiv preprint arXiv:1806.03536*, 2018.

[196] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[197] Jianwei Yang, Zhile Ren, Mingze Xu, Xinlei Chen, David J. Crandall, Devi Parikh, and Dhruv Batra. Embodied Amodal Recognition: Learning to Move to Perceive Objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[198] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting Semi-Supervised Learning with Graph Embeddings. *33rd International Conference on Machine Learning, ICML 2016*, 1:86–94, 3 2016.

[199] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. Stacked Attention Networks for Image Question Answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 21–29, 2016.

[200] Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A Scalable Active Framework for Region Annotation in 3D Shape Collections. *ACM Transactions on Graphics (TOG)*, 35(6):1–12, 2016.

[201] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical Graph Representation Learning with Differentiable Pooling. In *Advances in Neural Information Processing Systems*, pages 4800–4810, 2018.

[202] Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian Model-Agnostic Meta-Learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 7343–7353, 2018.

[203] Licheng Yu, Xinlei Chen, Georgia Gkioxari, Mohit Bansal, Tamara L. Berg, and Dhruv Batra. Multi-Target Embodied Question Answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[204] Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, Murray Shanahan, Victoria Langston, Razvan Pascanu, Matthew Botvinick, Oriol Vinyals, and Peter Battaglia. Deep reinforcement learning with relational inductive biases. In *International Conference on Learning Representations*, 2019.

[205] Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. From Recognition to Cognition: Visual Commonsense Reasoning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[206] Min Zeng, Fuhao Zhang, Fang-Xiang Wu, Yaohang Li, Jianxin Wang, and Min Li. Protein–Protein Interaction Site Prediction through Combining Local and Global Features with Deep Neural Networks. *Bioinformatics*, September 2019.

[207] Jiawei Zhang. Get rid of suspended animation problem: Deep diffusive neural network on graph semi-supervised classification. *arXiv preprint arXiv:2001.07922*, 2020.

[208] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Proceedings of AAAI Conference on Artificial Inteligence*, 2018.

[209] Qi Zhao, Ze Ye, Chao Chen, and Yusu Wang. Persistence Enhanced Graph Neural Network. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, pages 2896–2906, 2020.

[210] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph Neural Networks: A Review of Methods and Applications. *arXiv e-prints*, art. arXiv:1812.08434, December 2018.

[211] Xiaojin Zhu and Zoubin Ghahramani. Learning from Labeled and Unlabeled Data with Label Propagation. 2002.

[212] Daniel Zügner and Stephan Günnemann. *Certifiable Robustness and Robust Training for Graph Convolutional Networks*. Association for Computing Machinery, New York, NY, USA, 2019. ISBN 9781450362016.

Appendix A

Additional details and results

A.1 VideoNavQA question templates and respective counts

- **Equals**<**attr**>
  'Are all <attr> obj_type-pl> color>?': 4014
  'Are all <attr> <obj_type-pl> in the <room_type>?': 3811
  'Are all <attr> things <obj_type-pl>?': 3539
  'Are both the <attr1> <obj_type1> and the <attr2> <obj_type2> <color>?': 3968
  'Are both the <attr1> <obj_type1> and the <attr2> <obj_type2> in the <room_type>?': 3804
  'Are the <attr1> <obj_type1> and the <attr2> <obj_type2> the same color?': 4018
  'Is the <attr1> thing <rel> the <attr2> <obj_type2> <art> <obj_type1>?': 3315

- **Count**
  'How many <attr1> <obj_type1-pl> are in the room containing the <attr2> <obj_type2>?': 3999
  'How many <attr> <obj_type-pl> are in the <room_type>?': 3763
  'How many <obj_type-pl> are <attr>?': 4120
  'How many rooms have <attr> <obj_type-pl>?': 3834

- **Compare**<**count**>
  'Are there <comp> <attr1> <obj_type1-pl> than <attr2> <obj_type2-pl>?': 4058
  'Are there as many <attr1> <obj_type1-pl> as there are <attr2> <obj_type2-pl>?': 4100

- **Compare**<**size**>
  'Is the <attr1> <obj_type> <comp_rel> than the <attr2> one?': 3272
  'Is the <room_type1> <comp_rel> than the <room_type2>?': 3148

- **Exist**
  'Is there <art> <attr> <obj_type>?': 4122
  'Is there <art> <room_type>?': 3335
  'Is there a room that has set(<art> <attr{}> <obj_type{}>)?': 3877
  'Is there set(<art> <attr{}> <obj_type{}>) in the <room_type>?': 4025
  'Is there set(<art> <attr{}> <obj_type{}>)?': 4107
  'Is there set(<art> <room_type{}>)?': 3750

- **Query<color>**
  'What color is the <attr1> <obj_type1> <rel> the <attr2> <obj_type2>?': 2178
  'What color is the <attr> <obj_type>?': 3592

- **Query<obj_type>**
  'What is the <attr1> thing <rel> the <attr2> <obj_type2>?': 3119
  'What is the <attr> thing?': 2883

- **Query<room_location>**
  'Where are the set(<attr{}> <obj_type{}>)?': 3816
  'Where is the <attr1> <obj_type1> <rel> the <attr2> <obj_type2>?': 2284
  'Where is the <attr> <obj_type>?': 3481

A.2 VideoNavQA model hyperparameters

In order to find the best performance on VideoNavQA, we ran several combinations of hyperparameters for each of the described models. I detail the model configurations that were evaluated on the validation set below. The average running time per epoch for the visual reasoning models was approximately 5 hours on a 16GB Tesla P100 GPU.

LSTM Embedding size: {128, 256, 512, 1024}. Learning rate: $\{5e^{-5}, 1e^{-4}\}$.

BoW Embedding size: {128, 256, 512}. Learning rate: $\{1e^{-5}, 5e^{-5}\}$.

FiLM GP Number of ResBlocks: {3, 4, 5}. Learning rate: $\{1e^{-4}, 1e^{-3}\}$. Number of classifier channels: {32, 64}.

FiLM AT Number of ResBlocks: {3, 4, 5}. Attention hidden size: {128, 256}. Learning rate: $\{1e^{-5}, 1e^{-4}\}$.

Multi-hop Number of ResBlocks: {3, 4, 5}. Learning rate: $\{1e^{-5}, 1e^{-4}\}$. Number of classifier channels: {32, 64}.

MAC Number of CNN Layers: {2,3}. Width: {512, 1024}. MAC time steps: {5,6}. We adapt the ramp-up/down Adam learning schedule popularly used in VQA [24],

ramping up the learning rate to $1e^{-4}$ in the first 2 epochs and then decaying it to $1e^{-5}$ after epoch 10 (training is done for a total of 15 epochs).

A.3 Pooling hyperparameters

As part of the experiments for the Mapper PageRank pooling operator described in Chapter 4, I additionally performed a hyperparameter search for DiffPool on hidden sizes $32, 64, 128$ and for MPR, over the following sets of possible values:

- all datasets: cover sizes $\{[40, 10], [20, 5]\}$, interval overlap $\{10\%, 25\%\}$;
- D&D: learning rate $\{5e^{-4}, 1e^{-3}\}$;
- Proteins: learning rate $\{2e^{-4}, 5e^{-4}, 1e^{-3}\}$, cover sizes $\{[24, 6], [16, 4], [12, 3], [8, 2]\}$, hidden sizes $\{64, 128\}$.

A.4 More results from the MPNP evaluation

Method	Accuracy %		F-measure		MCC	
ISIS	69.4		0.267		0.097	
DeepPPISP	65.5		**0.397**		0.206	
R-GCN	76.7		0.165		0.169	
	5	30	5	30	5	30
NP	77.5	79.3	0.212	0.180	0.145	0.150
R-MPNP	**79.1**	**80.7**	0.292	0.348	**0.236**	**0.284**

Table A.1: Node classification on Protein-Protein Interaction Site Prediction. R-MPNP scores for $\{5, 30\}$% sampling rates. Results for ISIS, DeepPPISP and R-GCN are taken from Ofran and Rost [142], Zeng et al. [206], and Schlichtkrull et al. [161], respectively.

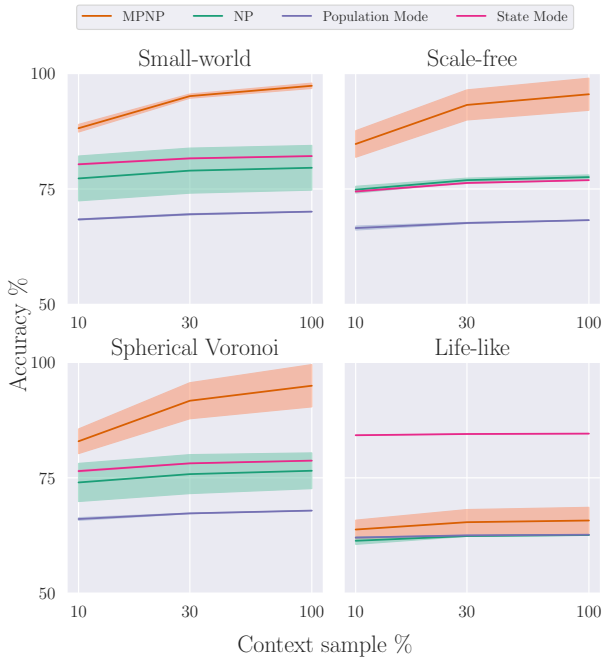

Figure A.1: State evolution accuracy $\pm\sigma$ for density- and count-based cellular automata. Models are trained at 30-50% context sampling. Testing at 100% effectively judges the quality of the rule embedding under perfect information.

#	Model	1%	5%	10%	30%
3	NP-c	67.00 ± 1.83	76.99 ± 1.50	78.56 ± 1.19	79.61 ± 1.20
	MPNP-c	**79.71 ± 1.04**	**88.28 ± 0.59**	**89.41 ± 0.58**	**90.02 ± 0.60**
	LP	65.31 ± 0.73	75.57 ± 0.31	77.90 ± 0.16	82.04 ± 0.18
	Mode	54.35 ± 0.10	54.28 ± 0.07	54.40 ± 0.27	54.41 ± 0.18
7	NP-c	52.83 ± 0.49	63.02 ± 0.50	64.29 ± 0.43	65.23 ± 0.51
	MPNP-c	**58.40 ± 0.77**	**68.96 ± 1.08**	**70.53 ± 0.88**	71.54 ± 0.91
	LP	52.62 ± 0.31	64.85 ± 0.22	68.55 ± 0.14	**74.96 ± 0.20**
	Mode	30.48 ± 0.16	30.57 ± 0.07	30.50 ± 0.10	30.50 ± 0.10
11	NP-c	34.57 ± 2.18	37.94 ± 0.84	38.88 ± 0.80	39.42 ± 0.78
	MPNP-c	43.62 ± 1.01	50.64 ± 1.14	51.87 ± 1.23	52.67 ± 1.24
	LP	**46.84 ± 0.55**	**60.11 ± 0.12**	**64.22 ± 0.08**	**71.73 ± 0.05**
	Mode	21.60 ± 0.08	21.60 ± 0.11	21.63 ± 0.09	21.66 ± 0.10

Table A.2: Results on the Cora-Branched transductive learning tasks for 3, 7 and 11 classes (#). Mean accuracy and standard deviations are reported at $\{1, 5, 10, 30\}$% context points.

#	Model	1%	5%	10%
2	NP-c	59.25	63.53	64.29
	MPNP-c	**62.91**	**67.53**	**68.57**
3	NP-c	49.82	56.93	59.03
	MPNP-c	**53.83**	**63.75**	**64.52**
5	NP-c	36.84	42.68	44.10
	MPNP-c	**41.67**	**49.99**	**51.15**
11	NP-c	19.71	21.13	21.82
	MPNP-c	**23.56**	**26.00**	**27.44**

Table A.3: Performance on the Cora-Branched few-shot learning generalisation tasks for 2, 3, 5 and 11 class $(\#)$ tasks. Accuracy at $\{1, 5, 10\}$% context points.

A.5 Numerical results for ShapeNet

	0.1%	1%	5%	10%	30%
Bag	71.08 ± 3.78	75.12 ± 1.44	75.57 ± 0.89	76.05 ± 0.13	73.21 ± 0.72
Cap	64.00 ± 4.28	68.76 ± 4.32	73.05 ± 0.92	69.42 ± 1.34	67.41 ± 0.47
Knife	79.82 ± 0.25	87.34 ± 1.47	89.93 ± 0.46	90.39 ± 0.34	90.34 ± 0.24
Laptop	90.39 ± 0.31	95.94 ± 0.17	96.71 ± 0.19	96.75 ± 0.00	97.07 ± 0.12
Mug	75.90 ± 1.37	85.63 ± 2.27	87.80 ± 1.02	88.70 ± 1.20	88.14 ± 0.02
Earphone	49.80 ± 4.45	57.14 ± 1.99	59.41 ± 1.44	55.59 ± 1.22	55.35 ± 0.70
Guitar	77.95 ± 0.61	89.12 ± 0.31	92.33 ± 0.25	92.75 ± 0.31	93.17 ± 0.11
Pistol	67.68 ± 0.95	82.51 ± 0.60	85.82 ± 0.29	85.57 ± 0.46	86.48 ± 0.16
Rocket	54.61 ± 0.21	56.03 ± 0.48	60.78 ± 0.74	64.26 ± 2.48	62.90 ± 0.04
Skateboard	41.67 ± 2.01	51.75 ± 0.76	55.10 ± 0.13	53.44 ± 1.05	52.33 ± 0.15
Table	75.19 ± 0.81	83.64 ± 0.30	85.80 ± 0.04	85.94 ± 0.01	86.61 ± 0.03
Airplane	58.32 ± 0.82	81.55 ± 0.16	86.68 ± 0.06	87.32 ± 0.05	87.90 ± 0.00
Car	43.08 ± 0.91	69.02 ± 1.45	76.56 ± 0.37	78.02 ± 0.12	78.53 ± 0.10
Chair	72.29 ± 0.00	86.73 ± 0.32	89.88 ± 0.26	90.22 ± 0.00	90.70 ± 0.03
Lamp	61.80 ± 1.31	79.44 ± 0.00	84.03 ± 0.15	84.45 ± 0.25	84.89 ± 0.01
Motorbike	27.54 ± 1.36	48.10 ± 1.09	53.94 ± 0.16	53.17 ± 0.38	53.76 ± 0.02

Table A.4: Numerical mIoU results for the MPNP on ShapeNet single-category tasks ($\mu \pm \sigma$).

	0.1%	1%	5%	10%	30%
Bag	52.62 ± 0.00	54.20 ± 2.23	52.87 ± 0.35	53.06 ± 0.12	53.46 ± 0.13
Cap	45.08 ± 6.01	56.88 ± 0.60	55.21 ± 0.40	59.67 ± 1.18	57.94 ± 0.71
Knife	72.84 ± 0.33	87.02 ± 0.65	89.49 ± 0.12	89.68 ± 0.31	89.12 ± 0.20
Laptop	82.42 ± 2.05	93.49 ± 0.24	96.07 ± 0.38	96.46 ± 0.12	96.72 ± 0.02
Mug	68.52 ± 1.71	80.95 ± 0.41	84.92 ± 0.61	84.58 ± 0.97	85.57 ± 0.01
Earphone	36.42 ± 7.28	47.98 ± 1.06	47.94 ± 0.68	47.79 ± 0.04	49.04 ± 0.24
Guitar	69.00 ± 0.36	83.41 ± 0.64	87.83 ± 0.50	88.12 ± 0.16	89.11 ± 0.01
Pistol	63.84 ± 2.02	70.42 ± 0.79	70.64 ± 0.15	71.94 ± 0.22	71.79 ± 0.17
Rocket	56.71 ± 2.41	59.76 ± 0.69	63.69 ± 0.54	62.50 ± 0.58	63.85 ± 0.31
Skateboard	32.13 ± 1.71	41.84 ± 0.07	40.78 ± 0.03	40.36 ± 0.19	40.25 ± 0.09
Table	76.53 ± 0.21	83.11 ± 0.08	84.18 ± 0.08	84.20 ± 0.12	84.26 ± 0.01
Airplane	44.92 ± 0.06	78.05 ± 0.22	83.05 ± 0.02	83.65 ± 0.02	84.04 ± 0.05
Car	38.86 ± 0.43	57.90 ± 1.33	63.89 ± 0.08	64.73 ± 0.14	65.55 ± 0.46
Chair	69.68 ± 1.14	84.78 ± 0.51	87.35 ± 0.10	87.69 ± 0.11	87.81 ± 0.01
Lamp	57.04 ± 1.73	71.88 ± 0.54	75.40 ± 0.45	75.49 ± 0.19	76.19 ± 0.01
Motorbike	21.28 ± 1.41	25.44 ± 0.05	25.66 ± 0.04	25.69 ± 0.16	25.73 ± 0.05

Table A.5: Numerical mIoU results for the NP on ShapeNet single-category tasks ($\mu \pm \sigma$).

	GCN	`labelprop`				
		0.1%	1%	5%	10%	30%
Bag	69.76	54.62 ± 1.88	70.10 ± 4.35	86.16 ± 1.05	90.45 ± 1.10	95.67 ± 0.54
Cap	65.85	47.76 ± 5.07	74.19 ± 3.53	84.97 ± 0.49	88.83 ± 0.62	93.43 ± 0.41
Knife	79.61	57.48 ± 3.40	88.82 ± 0.56	93.70 ± 0.36	95.01 ± 0.24	97.03 ± 0.10
Laptop	94.23	58.61 ± 2.64	88.16 ± 0.64	93.76 ± 0.17	95.46 ± 0.12	97.34 ± 0.05
Mug	85.40	47.44 ± 1.48	74.93 ± 2.42	88.15 ± 0.45	91.09 ± 0.71	94.19 ± 0.15
Earphone	49.76	36.35 ± 2.29	66.68 ± 1.96	78.45 ± 1.21	82.50 ± 0.73	88.24 ± 0.27
Guitar	89.01	37.85 ± 1.69	78.79 ± 1.97	92.06 ± 0.20	94.10 ± 0.20	96.44 ± 0.09
Pistol	76.72	39.80 ± 1.22	69.09 ± 1.56	83.25 ± 0.94	87.02 ± 0.23	92.39 ± 0.21
Rocket	56.31	38.43 ± 3.14	59.84 ± 7.51	80.95 ± 0.52	85.67 ± 1.15	91.53 ± 0.37
Skateboard	57.19	32.83 ± 1.57	56.59 ± 1.35	80.33 ± 0.71	84.91 ± 0.78	90.76 ± 0.35
Table	76.54	42.22 ± 0.53	68.88 ± 0.27	83.32 ± 0.08	86.83 ± 0.06	91.16 ± 0.07
Airplane	79.50	20.81 ± 0.21	60.48 ± 0.33	79.77 ± 0.13	84.50 ± 0.07	90.47 ± 0.04
Car	71.95	19.20 ± 0.38	45.91 ± 0.99	67.85 ± 0.38	75.33 ± 0.31	85.36 ± 0.05
Chair	77.84	33.06 ± 0.62	70.97 ± 0.27	87.04 ± 0.12	90.65 ± 0.07	94.65 ± 0.02
Lamp	53.78	58.61 ± 2.64	88.16 ± 0.64	93.76 ± 0.17	95.46 ± 0.12	97.34 ± 0.05
Motorbike	46.18	15.66 ± 1.14	38.46 ± 1.28	63.11 ± 1.46	74.05 ± 0.51	85.51 ± 0.36

Table A.6: Numerical mIoU results for GCN and `labelprop` on ShapeNet single-category tasks ($\mu \pm \sigma$). Note that the GCN does not use the context labels and thus produces deterministic outputs.