Relational inductive biases, deep learning, and graph networks

Peter W. Battaglia[1]*, **Jessica B. Hamrick**[1], **Victor Bapst**[1], **Alvaro Sanchez-Gonzalez**[1], **Vinicius Zambaldi**[1], **Mateusz Malinowski**[1], **Andrea Tacchetti**[1], **David Raposo**[1], **Adam Santoro**[1], **Ryan Faulkner**[1], **Caglar Gulcehre**[1], **Francis Song**[1], **Andrew Ballard**[1], **Justin Gilmer**[2], **George Dahl**[2], **Ashish Vaswani**[2], **Kelsey Allen**[3], **Charles Nash**[4], **Victoria Langston**[1], **Chris Dyer**[1], **Nicolas Heess**[1], **Daan Wierstra**[1], **Pushmeet Kohli**[1], **Matt Botvinick**[1], **Oriol Vinyals**[1], **Yujia Li**[1], **Razvan Pascanu**[1]

[1]DeepMind; [2]Google Brain; [3]MIT; [4]University of Edinburgh

Abstract

Artificial intelligence (AI) has undergone a renaissance recently, making major progress in key domains such as vision, language, control, and decision-making. This has been due, in part, to cheap data and cheap compute resources, which have fit the natural strengths of deep learning. However, many defining characteristics of human intelligence, which developed under much different pressures, remain out of reach for current approaches. In particular, generalizing beyond one's experiences—a hallmark of human intelligence from infancy—remains a formidable challenge for modern AI.

The following is part position paper, part review, and part unification. We argue that combinatorial generalization must be a top priority for AI to achieve human-like abilities, and that structured representations and computations are key to realizing this objective. Just as biology uses nature and nurture cooperatively, we reject the false choice between "hand-engineering" and "end-to-end" learning, and instead advocate for an approach which benefits from their complementary strengths. We explore how using *relational inductive biases* within deep learning architectures can facilitate learning about entities, relations, and rules for composing them. We present a new building block for the AI toolkit with a strong relational inductive bias—the *graph network*—which generalizes and extends various approaches for neural networks that operate on graphs, and provides a straightforward interface for manipulating structured knowledge and producing structured behaviors. We discuss how graph networks can support relational reasoning and combinatorial generalization, laying the foundation for more sophisticated, interpretable, and flexible patterns of reasoning.

1 Introduction

A key signature of human intelligence is the ability to make "infinite use of finite means" (Humboldt, 1836; Chomsky, 1965), in which a small set of elements (such as words) can be productively composed in limitless ways (such as into new sentences). This reflects the principle of *combinatorial generalization*, that is, constructing new inferences, predictions, and behaviors from known building blocks. Here we explore how to improve modern AI's capacity for combinatorial generalization by biasing learning towards structured representations and computations, and in particular, systems that operate on graphs.

*Corresponding author: peterbattaglia@google.com

Humans' capacity for combinatorial generalization depends critically on our cognitive mechanisms for representing structure and reasoning about relations. We represent complex systems as compositions of entities and their interactions[1] (Navon, 1977; McClelland and Rumelhart, 1981; Plaut et al., 1996; Marcus, 2001; Goodwin and Johnson-Laird, 2005; Kemp and Tenenbaum, 2008), such as judging whether a haphazard stack of objects is stable (Battaglia et al., 2013). We use hierarchies to abstract away from fine-grained differences, and capture more general commonalities between representations and behaviors (Botvinick, 2008; Tenenbaum et al., 2011), such as parts of an object, objects in a scene, neighborhoods in a town, and towns in a country. We solve novel problems by composing familiar skills and routines (Anderson, 1982), for example traveling to a new location by composing familiar procedures and objectives, such as "travel by airplane", "to San Diego", "eat at", and "an Indian restaurant". We draw analogies by aligning the relational structure between two domains and drawing inferences about one based on corresponding knowledge about the other (Gentner and Markman, 1997; Hummel and Holyoak, 2003).

Kenneth Craik's "The Nature of Explanation" (1943), connects the compositional structure of the world to how our internal mental models are organized:

> *...[a human mental model] has a similar relation-structure to that of the process it imitates. By 'relation-structure' I do not mean some obscure non-physical entity which attends the model, but the fact that it is a working physical model which works in the same way as the process it parallels... physical reality is built up, apparently, from a few fundamental types of units whose properties determine many of the properties of the most complicated phenomena, and this seems to afford a sufficient explanation of the emergence of analogies between mechanisms and similarities of relation-structure among these combinations without the necessity of any theory of objective universals.* (Craik, 1943, page 51-55)

That is, the world is compositional, or at least, we understand it in compositional terms. When learning, we either fit new knowledge into our existing structured representations, or adjust the structure itself to better accommodate (and make use of) the new and the old (Tenenbaum et al., 2006; Griffiths et al., 2010; Ullman et al., 2017).

The question of how to build artificial systems which exhibit combinatorial generalization has been at the heart of AI since its origins, and was central to many structured approaches, including logic, grammars, classic planning, graphical models, causal reasoning, Bayesian nonparametrics, and probabilistic programming (Chomsky, 1957; Nilsson and Fikes, 1970; Pearl, 1986, 2009; Russell and Norvig, 2009; Hjort et al., 2010; Goodman et al., 2012; Ghahramani, 2015). Entire sub-fields have focused on explicit entity- and relation-centric learning, such as relational reinforcement learning (Džeroski et al., 2001) and statistical relational learning (Getoor and Taskar, 2007). A key reason why structured approaches were so vital to machine learning in previous eras was, in part, because data and computing resources were expensive, and the improved sample complexity afforded by structured approaches' strong inductive biases was very valuable.

In contrast with past approaches in AI, modern deep learning methods (LeCun et al., 2015; Schmidhuber, 2015; Goodfellow et al., 2016) often follow an "end-to-end" design philosophy which emphasizes minimal *a priori* representational and computational assumptions, and seeks to avoid explicit structure and "hand-engineering". This emphasis has fit well with—and has perhaps been affirmed by—the current abundance of cheap data and cheap computing resources, which make trading off sample efficiency for more flexible learning a rational choice. The remarkable and rapid advances across many challenging domains, from image classification (Krizhevsky et al., 2012;

[1] Whether this entails a "language of thought" (Fodor, 1975) is beyond the scope of this work.

Szegedy et al., 2017), to natural language processing (Sutskever et al., 2014; Bahdanau et al., 2015), to game play (Mnih et al., 2015; Silver et al., 2016; Moravčík et al., 2017), are a testament to this minimalist principle. A prominent example is from language translation, where sequence-to-sequence approaches (Sutskever et al., 2014; Bahdanau et al., 2015) have proven very effective without using explicit parse trees or complex relationships between linguistic entities.

Despite deep learning's successes, however, important critiques (Marcus, 2001; Shalev-Shwartz et al., 2017; Lake et al., 2017; Lake and Baroni, 2018; Marcus, 2018a,b; Pearl, 2018; Yuille and Liu, 2018) have highlighted key challenges it faces in complex language and scene understanding, reasoning about structured data, transferring learning beyond the training conditions, and learning from small amounts of experience. These challenges demand combinatorial generalization, and so it is perhaps not surprising that an approach which eschews compositionality and explicit structure struggles to meet them.

When deep learning's connectionist (Rumelhart et al., 1987) forebears were faced with analogous critiques from structured, symbolic positions (Fodor and Pylyshyn, 1988; Pinker and Prince, 1988), there was a constructive effort (Bobrow and Hinton, 1990; Marcus, 2001) to address the challenges directly and carefully. A variety of innovative sub-symbolic approaches for representing and reasoning about structured objects were developed in domains such as analogy-making, linguistic analysis, symbol manipulation, and other forms of relational reasoning (Smolensky, 1990; Hinton, 1990; Pollack, 1990; Elman, 1991; Plate, 1995; Eliasmith, 2013), as well as more integrative theories for how the mind works (Marcus, 2001). Such work also helped cultivate more recent deep learning advances which use distributed, vector representations to capture rich semantic content in text (Mikolov et al., 2013; Pennington et al., 2014), graphs (Narayanan et al., 2016, 2017), algebraic and logical expressions (Allamanis et al., 2017; Evans et al., 2018), and programs (Devlin et al., 2017; Chen et al., 2018b).

We suggest that a key path forward for modern AI is to commit to combinatorial generalization as a top priority, and we advocate for integrative approaches to realize this goal. Just as biology does not choose between nature *versus* nurture—it uses nature and nurture *jointly*, to build wholes which are greater than the sums of their parts—we, too, reject the notion that structure and flexibility are somehow at odds or incompatible, and embrace both with the aim of reaping their complementary strengths. In the spirit of numerous recent examples of principled hybrids of structure-based methods and deep learning (e.g., Reed and De Freitas, 2016; Garnelo et al., 2016; Ritchie et al., 2016; Wu et al., 2017; Denil et al., 2017; Hudson and Manning, 2018), we see great promise in synthesizing new techniques by drawing on the full AI toolkit and marrying the best approaches from today with those which were essential during times when data and computation were at a premium.

Recently, a class of models has arisen at the intersection of deep learning and structured approaches, which focuses on approaches for reasoning about explicitly structured data, in particular graphs (e.g. Scarselli et al., 2009b; Bronstein et al., 2017; Gilmer et al., 2017; Wang et al., 2018c; Li et al., 2018; Kipf et al., 2018; Gulcehre et al., 2018). What these approaches all have in common is a capacity for performing computation over discrete entities and the relations between them. What sets them apart from classical approaches is how the representations and structure of the entities and relations—and the corresponding computations—can be learned, relieving the burden of needing to specify them in advance. Crucially, these methods carry strong *relational inductive biases*, in the form of specific architectural assumptions, which guide these approaches towards learning about entities and relations (Mitchell, 1980), which we, joining many others (Spelke et al., 1992; Spelke and Kinzler, 2007; Marcus, 2001; Tenenbaum et al., 2011; Lake et al., 2017; Lake and Baroni, 2018; Marcus, 2018b), suggest are an essential ingredient for human-like intelligence.

In the remainder of the paper, we examine various deep learning methods through the lens of their relational inductive biases, showing that existing methods often carry relational assumptions

Box 1: Relational reasoning

We define *structure* as the product of composing a set of known building blocks. "Structured representations" capture this composition (i.e., the arrangement of the elements) and "structured computations" operate over the elements and their composition as a whole. Relational reasoning, then, involves manipulating structured representations of *entities* and *relations*, using *rules* for how they can be composed. We use these terms to capture notions from cognitive science, theoretical computer science, and AI, as follows:

- An *entity* is an element with attributes, such as a physical object with a size and mass.
- A *relation* is a property between entities. Relations between two objects might include SAME SIZE AS, HEAVIER THAN, and DISTANCE FROM. Relations can have attributes as well. The relation MORE THAN X TIMES HEAVIER THAN takes an attribute, X, which determines the relative weight threshold for the relation to be TRUE vs. FALSE. Relations can also be sensitive to the global context. For a stone and a feather, the relation FALLS WITH GREATER ACCELERATION THAN depends on whether the context is IN AIR vs. IN A VACUUM. Here we focus on pairwise relations between entities.
- A *rule* is a function (like a non-binary logical predicate) that maps entities and relations to other entities and relations, such as a scale comparison like IS ENTITY X LARGE? and IS ENTITY X HEAVIER THAN ENTITY Y?. Here we consider rules which take one or two arguments (unary and binary), and return a unary property value.

As an illustrative example of relational reasoning in machine learning, graphical models (Pearl, 1988; Koller and Friedman, 2009) can represent complex joint distributions by making explicit random conditional independences among random variables. Such models have been very successful because they capture the sparse structure which underlies many real-world generative processes and because they support efficient algorithms for learning and reasoning. For example, hidden Markov models constrain latent states to be conditionally independent of others given the state at the previous time step, and observations to be conditionally independent given the latent state at the current time step, which are well-matched to the relational structure of many real-world causal processes. Explicitly expressing the sparse dependencies among variables provides for various efficient inference and reasoning algorithms, such as message-passing, which apply a common information propagation procedure across localities within a graphical model, resulting in a composable, and partially parallelizable, reasoning procedure which can be applied to graphical models of different sizes and shape.

which are not always explicit or immediately evident. We then present a general framework for entity- and relation-based reasoning—which we term *graph networks*—for unifying and extending existing methods which operate on graphs, and describe key design principles for building powerful architectures using graph networks as building blocks.

2 Relational inductive biases

Many approaches in machine learning and AI which have a capacity for relational reasoning (Box 1) use a *relational inductive bias*. While not a precise, formal definition, we use this term to refer generally to inductive biases (Box 2) which impose constraints on relationships and interactions among entities in a learning process.

Box 2: Inductive biases

Learning is the process of apprehending useful knowledge by observing and interacting with the world. It involves searching a space of solutions for one expected to provide a better explanation of the data or to achieve higher rewards. But in many cases, there are multiple solutions which are equally good (Goodman, 1955). An *inductive bias* allows a learning algorithm to prioritize one solution (or interpretation) over another, independent of the observed data (Mitchell, 1980). In a Bayesian model, inductive biases are typically expressed through the choice and parameterization of the prior distribution (Griffiths et al., 2010). In other contexts, an inductive bias might be a regularization term (McClelland, 1994) added to avoid overfitting, or it might be encoded in the architecture of the algorithm itself. Inductive biases often trade flexibility for improved sample complexity and can be understood in terms of the bias-variance tradeoff (Geman et al., 1992). Ideally, inductive biases both improve the search for solutions without substantially diminishing performance, as well as help find solutions which generalize in a desirable way; however, mismatched inductive biases can also lead to suboptimal performance by introducing constraints that are too strong.

Inductive biases can express assumptions about either the data-generating process or the space of solutions. For example, when fitting a 1D function to data, linear least squares follows the constraint that the approximating function be a linear model, and approximation errors should be minimal under a quadratic penalty. This reflects an assumption that the data-generating process can be explained simply, as a line process corrupted by additive Gaussian noise. Similarly, $L2$ regularization prioritizes solutions whose parameters have small values, and can induce unique solutions and global structure to otherwise ill-posed problems. This can be interpreted as an assumption about the learning process: that searching for good solutions is easier when there is less ambiguity among solutions. Note, these assumptions need not be explicit—they reflect interpretations of how a model or algorithm interfaces with the world.

Creative new machine learning architectures have rapidly proliferated in recent years, with (perhaps not surprisingly given the thesis of this paper) practitioners often following a design pattern of composing elementary building blocks to form more complex, deep[2] computational hierarchies and graphs[3]. Building blocks such as "fully connected" layers are stacked into "multilayer perceptrons" (MLPs), "convolutional layers" are stacked into "convolutional neural networks" (CNNs), and a standard recipe for an image processing network is, generally, some variety of CNN composed with a MLP. This composition of layers provides a particular type of relational inductive bias—that of hierarchical processing—in which computations are performed in stages, typically resulting in increasingly long range interactions among information in the input signal. As we explore below, the building blocks themselves also carry various relational inductive biases (Table 1). Though beyond the scope of this paper, various non-relational inductive biases are used in deep learning as well: for example, activation non-linearities, weight decay, dropout (Srivastava et al., 2014), batch and layer normalization (Ioffe and Szegedy, 2015; Ba et al., 2016), data augmentation, training curricula, and optimization algorithms all impose constraints on the trajectory and outcome of learning.

To explore the relational inductive biases expressed within various deep learning methods, we must identify several key ingredients, analogous to those in Box 1: what are the *entities*, what are the *relations*, and what are the *rules* for composing entities and relations, and computing their

[2]This pattern of composition in depth is ubiquitous in deep learning, and is where the "deep" comes from.

[3]Recent methods (Liu et al., 2018) even automate architecture construction via learned graph editing procedures.

Component	Entities	Relations	Rel. inductive bias	Invariance
Fully connected	Units	All-to-all	Weak	-
Convolutional	Grid elements	Local	Locality	Spatial translation
Recurrent	Timesteps	Sequential	Sequentiality	Time translation
Graph network	Nodes	Edges	Arbitrary	Node, edge permutations

Table 1: Various relational inductive biases in standard deep learning components. See also Section 2.

implications? In deep learning, the entities and relations are typically expressed as distributed representations, and the rules as neural network function approximators; however, the precise forms of the entities, relations, and rules vary between architectures. To understand these differences between architectures, we can further ask how each supports relational reasoning by probing:

- The *arguments* to the rule functions (e.g., which entities and relations are provided as input).
- How the rule function is *reused*, or *shared*, across the computational graph (e.g., across different entities and relations, across different time or processing steps, etc.).
- How the architecture defines *interactions* versus *isolation* among representations (e.g., by applying rules to draw conclusions about related entities, versus processing them separately).

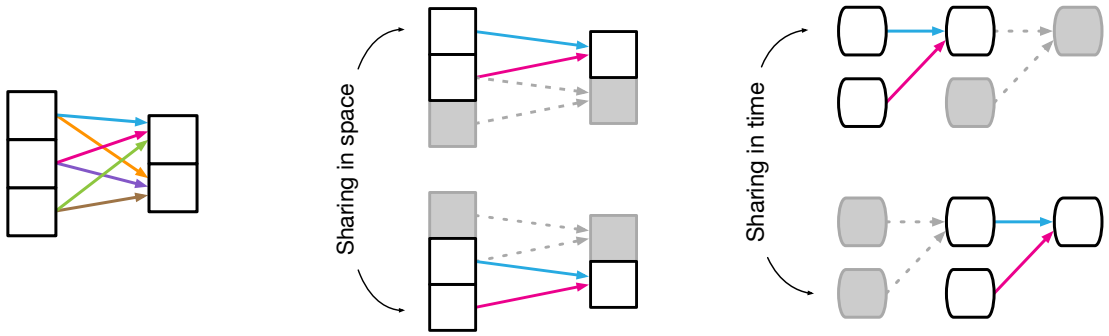
2.1 Relational inductive biases in standard deep learning building blocks

2.1.1 Fully connected layers

Perhaps the most common building block is a fully connected layer (Rosenblatt, 1961). Typically implemented as a non-linear vector-valued function of vector inputs, each element, or "unit", of the output vector is the dot product between a weight vector, followed by an added bias term, and finally a non-linearity such as a rectified linear unit (ReLU). As such, the entities are the units in the network, the relations are all-to-all (all units in layer i are connected to all units in layer j), and the rules are specified by the weights and biases. The argument to the rule is the full input signal, there is no reuse, and there is no isolation of information (Figure 1a). The implicit relational inductive bias in a fully connected layer is thus very weak: all input units can interact to determine any output unit's value, independently across outputs (Table 1).

2.1.2 Convolutional layers

Another common building block is a convolutional layer (Fukushima, 1980; LeCun et al., 1989). It is implemented by convolving an input vector or tensor with a kernel of the same rank, adding a bias term, and applying a point-wise non-linearity. The entities here are still individual units (or grid elements, e.g. pixels), but the relations are sparser. The differences between a fully connected layer and a convolutional layer impose some important relational inductive biases: locality and translation invariance (Figure 1b). Locality reflects that the arguments to the relational rule are those entities in close proximity with one another in the input signal's coordinate space, isolated from distal entities. Translation invariance reflects reuse of the same rule across localities in the input. These biases are very effective for processing natural image data because there is high covariance within local neighborhoods, which diminishes with distance, and because the statistics are mostly stationary across an image (Table 1).



(a) Fully connected (b) Convolutional (c) Recurrent

Figure 1: Reuse and sharing in common deep learning building blocks. (a) Fully connected layer, in which all weights are independent, and there is no sharing. (b) Convolutional layer, in which a local kernel function is reused multiple times across the input. Shared weights are indicated by arrows with the same color. (c) Recurrent layer, in which the same function is reused across different processing steps.

2.1.3 Recurrent layers

A third common building block is a recurrent layer (Elman, 1990), which is implemented over a sequence of steps. Here, we can view the inputs and hidden states at each processing step as the entities, and the Markov dependence of one step's hidden state on the previous hidden state and the current input, as the relations. The rule for combining the entities takes a step's inputs and hidden state as arguments to update the hidden state. The rule is reused over each step (Figure 1c), which reflects the relational inductive bias of temporal invariance (similar to a CNN's translational invariance in space). For example, the outcome of some physical sequence of events should not depend on the time of day. RNNs also carry a bias for locality in the sequence via their Markovian structure (Table 1).

2.2 Computations over sets and graphs

While the standard deep learning toolkit contains methods with various forms of relational inductive biases, there is no "default" deep learning component which operates on arbitrary relational structure. We need models with explicit representations of entities and relations, and learning algorithms which find rules for computing their interactions, as well as ways of grounding them in data. Importantly, entities in the world (such as objects and agents) do not have a natural order; rather, orderings can be defined by the properties of their relations. For example, the relations between the sizes of a set of objects can potentially be used to order them, as can their masses, ages, toxicities, and prices. Invariance to ordering—except in the face of relations—is a property that should ideally be reflected by a deep learning component for relational reasoning.

Sets are a natural representation for systems which are described by entities whose order is undefined or irrelevant; in particular, their relational inductive bias does not come from the *presence* of something, but rather from the *absence*. For illustration, consider the task of predicting the center of mass of a solar system comprised of n planets, whose attributes (e.g., mass, position, velocity, etc.) are denoted by $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$. For such a computation, the order in which we consider the planets does not matter because the state can be described solely in terms of aggregated, averaged

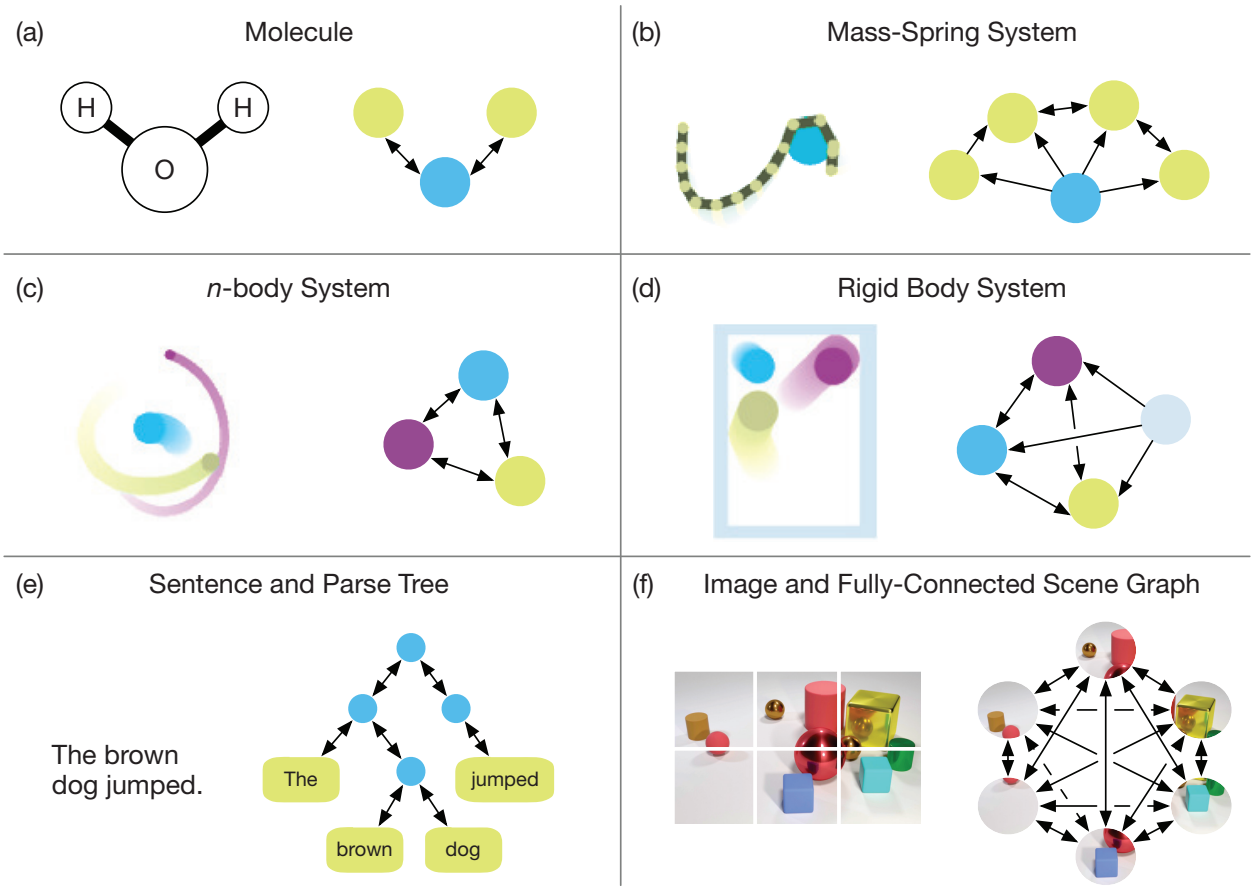


Figure 2: Different graph representations. (a) A molecule, in which each atom is represented as a node and edges correspond to bonds (e.g. Duvenaud et al., 2015). (b) A mass-spring system, in which the rope is defined by a sequence of masses which are represented as nodes in the graph (e.g. Battaglia et al., 2016; Chang et al., 2017). (c) A n-body system, in which the bodies are nodes and the underlying graph is fully connected (e.g. Battaglia et al., 2016; Chang et al., 2017). (d) A rigid body system, in which the balls and walls are nodes, and the underlying graph defines interactions between the balls and between the balls and the walls (e.g. Battaglia et al., 2016; Chang et al., 2017). (e) A sentence, in which the words correspond to leaves in a tree, and the other nodes and edges could be provided by a parser (e.g. Socher et al., 2013). Alternately, a fully connected graph could be used (e.g. Vaswani et al., 2017). (f) An image, which can be decomposed into image patches corresponding to nodes in a fully connected graph (e.g. Santoro et al., 2017; Wang et al., 2018c).

quantities. However, if we were to use a MLP for this task, having learned the prediction for a particular input $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ would not necessarily transfer to making a prediction for the same inputs under a different ordering $(\mathbf{x}_n, \mathbf{x}_1, \ldots, \mathbf{x}_2)$. Since there are $n!$ such possible permutations, in the worst case, the MLP could consider each ordering as fundamentally different, and thus require an exponential number of input/output training examples to learn an approximating function. A natural way to handle such combinatorial explosion is to only allow the prediction to depend on symmetric functions of the inputs' attributes. This might mean computing shared per-object features $\{f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)\}$ which are then aggregated in a symmetric way (for example, by taking their mean). Such an approach is the essence of the Deep Sets model (Zaheer et al., 2017), which we explore further in Section 4.2.3.

Of course, permutation invariance is not the only important form of underlying structure in many problems. For example, each object in a set may be affected by pairwise interactions with the other objects in the set. In our planets scenario, consider now the task of predicting each individual planet's position after a time interval, Δt. In this case, using aggregated, averaged information is not enough because the movement of each planet depends on the forces the other planets are exerting on it. Instead, we could compute the state of each object as $\mathbf{x}_i' = f(\mathbf{x}_i, \sum_j g(\mathbf{x}_i, \mathbf{x}_j))$, where g could compute the force induced by the j-th planet on the i-th planet, and f could compute the future state of the i-th planet which results from the forces and dynamics. The fact that we use the same g everywhere is again a consequence of the global permutation invariance of the system; however, it also supports a different relational structure because g now takes two arguments rather than one.[4]

The above solar system examples illustrate two relational structures: one in which there are no relations, and one which consists of all pairwise relations. Many real-world systems (such as in Figure 2) have a relational structure somewhere in between these two extremes, however, with some pairs of entities possessing a relation and others lacking one. In our solar system example, if the system instead consists of the planets and their moons, one may be tempted to approximate it by neglecting the interactions between moons of different planets. In practice, this means computing interactions only between some pairs of objects, i.e. $x_i' = f(\mathbf{x}_i, \sum_{j \in \delta(i)} g(\mathbf{x}_i, \mathbf{x}_j))$, where $\delta(i) \subseteq \{1, \ldots, n\}$ is a neighborhood around node i. This corresponds to a graph, in that the i-th object only interacts with a subset of the other objects, described by its neighborhood. Note, the updated states still do not depend in the order in which we describe the neighborhood.[5]

Graphs, generally, are a representation which supports arbitrary (pairwise) relational structure, and computations over graphs afford a strong relational inductive bias beyond that which convolutional and recurrent layers can provide.

3 Graph networks

Neural networks that operate on graphs, and structure their computations accordingly, have been developed and explored extensively for more than a decade under the umbrella of "graph neural networks" (Gori et al., 2005; Scarselli et al., 2005, 2009a; Li et al., 2016), but have grown rapidly in scope and popularity in recent years. We survey the literature on these methods in the next sub-section (3.1). Then in the remaining sub-sections, we present our *graph networks* framework, which generalizes and extends several lines of work in this area.

3.1 Background

Models in the graph neural network family (Gori et al., 2005; Scarselli et al., 2005, 2009a; Li et al., 2016) have been explored in a diverse range of problem domains, across supervised, semi-supervised, unsupervised, and reinforcement learning settings. They have been effective at tasks thought to have rich relational structure, such as visual scene understanding tasks (Raposo et al., 2017; Santoro et al., 2017) and few-shot learning (Garcia and Bruna, 2018). They have also been used to learn the dynamics of physical systems (Battaglia et al., 2016; Chang et al., 2017; Watters et al., 2017; van Steenkiste et al., 2018; Sanchez-Gonzalez et al., 2018) and multi-agent systems (Sukhbaatar

[4] We could extend this same analysis to increasingly entangled structures that depend on relations among triplets (i.e., $g(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$), quartets, and so on. We note that if we restrict these functions to only operate on subsets of $\mathbf{x}_i$ which are spatially close, then we end back up with something resembling CNNs. In the most entangled sense, where there is a single relation function $g(\mathbf{x}_1, \ldots, \mathbf{x}_n)$, we end back up with a construction similar to a fully connected layer.

[5] The invariance which this model enforces is the invariance under isomorphism of the graph.

et al., 2016; Hoshen, 2017; Kipf et al., 2018), to reason about knowledge graphs (Bordes et al., 2013; Oñoro-Rubio et al., 2017; Hamaguchi et al., 2017), to predict the chemical properties of molecules (Duvenaud et al., 2015; Gilmer et al., 2017), to predict traffic on roads (Cui et al., 2018), to classify and segment videos (Wang et al., 2018c) and 3D meshes and point clouds (Wang et al., 2018d), to classify regions in images (Chen et al., 2018a), to perform semi-supervised text classification (Kipf and Welling, 2017), and in machine translation (Vaswani et al., 2017; Shaw et al., 2018; Gulcehre et al., 2018). They have been used within both model-free (Wang et al., 2018b) and model-based (Hamrick et al., 2017; Pascanu et al., 2017; Sanchez-Gonzalez et al., 2018) continuous control, for model-free reinforcement learning (Hamrick et al., 2018; Zambaldi et al., 2018), and for more classical approaches to planning (Toyer et al., 2017).

Many traditional computer science problems, which involve reasoning about discrete entities and structure, have also been explored with graph neural networks, such as combinatorial optimization (Bello et al., 2016; Nowak et al., 2017; Dai et al., 2017), boolean satisfiability (Selsam et al., 2018), program representation and verification (Allamanis et al., 2018; Li et al., 2016), modeling cellular automata and Turing machines (Johnson, 2017), and performing inference in graphical models (Yoon et al., 2018). Recent work has also focused on building generative models of graphs (Li et al., 2018; De Cao and Kipf, 2018; You et al., 2018; Bojchevski et al., 2018), and unsupervised learning of graph embeddings (Perozzi et al., 2014; Tang et al., 2015; Grover and Leskovec, 2016; García-Durán and Niepert, 2017).

The works cited above are by no means an exhaustive list, but provide a representative cross-section of the breadth of domains for which graph neural networks have proven useful. We point interested readers to a number of existing reviews which examine the body of work on graph neural networks in more depth. In particular, Scarselli et al. (2009a) provides an authoritative overview of early graph neural network approaches. Bronstein et al. (2017) provides an excellent survey of deep learning on non-Euclidean data, and explores graph neural nets, graph convolution networks, and related spectral approaches. Recently, Gilmer et al. (2017) introduced the message-passing neural network (MPNN), which unified various graph neural network and graph convolutional network approaches (Monti et al., 2017; Bruna et al., 2014; Henaff et al., 2015; Defferrard et al., 2016; Niepert et al., 2016; Kipf and Welling, 2017; Bronstein et al., 2017) by analogy to message-passing in graphical models. In a similar vein, Wang et al. (2018c) introduced the non-local neural network (NLNN), which unified various "self-attention"-style methods (Vaswani et al., 2017; Hoshen, 2017; Veličković et al., 2018) by analogy to methods from computer vision and graphical models for capturing long range dependencies in signals.

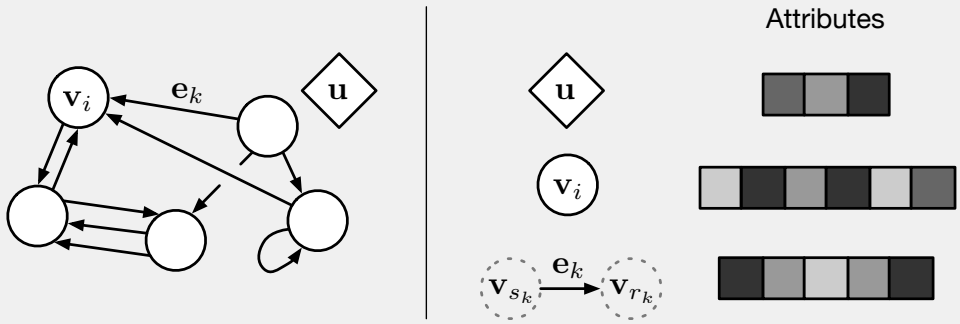
3.2 Graph network (GN) block

We now present our *graph networks* (GN) framework, which defines a class of functions for relational reasoning over graph-structured representations. Our GN framework generalizes and extends various graph neural network, MPNN, and NLNN approaches (Scarselli et al., 2009a; Gilmer et al., 2017; Wang et al., 2018c), and supports constructing complex architectures from simple building blocks.[6] Note, we avoided using the term "neural" in the "graph network" label to reflect that they can be implemented with functions other than neural networks, though here our focus is on neural network implementations.

The main unit of computation in the GN framework is the *GN block*, a "graph-to-graph" module which takes a graph as input, performs computations over the structure, and returns a graph as output. As described in Box 3, entities are represented by the graph's *nodes*, relations by the *edges*,

[6] We also plan to release an open-source library for graph networks later this year.

Box 3: Our definition of "graph"



Here we use "graph" to mean a directed, attributed multi-graph with a global attribute. In our terminology, a node is denoted as $\mathbf{v}_i$, an edge as $\mathbf{e}_k$, and the global attributes as $\mathbf{u}$. We also use s_k and r_k to indicate the indices of the sender and receiver nodes (see below), respectively, for edge k. To be more precise, we define these terms as:

Directed : one-way edges, from a "sender" node to a "receiver" node.
Attribute : properties that can be encoded as a vector, set, or even another graph.
Attributed : edges and vertices have attributes associated with them.
Global attribute : a graph-level attribute.
Multi-graph : there can be more than one edge between vertices, including self-edges.

Figure 2 shows a variety of different types of graphs corresponding to real data that we may be interested in modeling, including physical systems, molecules, images, and text.

and system-level properties by *global* attributes. The GN framework's block organization emphasizes customizability and synthesizing new architectures which express desired relational inductive biases. The key design principles are: *Flexible representations* (see Section 4.1); *Configurable within-block structure* (see Section 4.2); and *Composable multi-block architectures* (see Section 4.3).

We introduce a motivating example to help make the GN formalism more concrete. Consider predicting the movements a set of rubber balls in an arbitrary gravitational field, which, instead of bouncing against one another, each have one or more springs which connect them to some (or all) of the others. We will refer to this running example throughout the definitions below, to motivate the graph representation and the computations operating over it. Figure 2 depicts some other common scenarios that can be represented by graphs and reasoned over using graph networks.

3.2.1 Definition of "graph"

Within our GN framework, a *graph* is defined as a 3-tuple $G = (\mathbf{u}, V, E)$ (see Box 3 for details of graph representations). The $\mathbf{u}$ is a global attribute; for example, $\mathbf{u}$ might represent the gravitational field. The $V = \{\mathbf{v}_i\}_{i=1:N^v}$ is the set of nodes (of cardinality N^v), where each $\mathbf{v}_i$ is a node's attribute. For example, V might represent each ball, with attributes for position, velocity, and mass. The $E = \{(\mathbf{e}_k, r_k, s_k)\}_{k=1:N^e}$ is the set of edges (of cardinality N^e), where each $\mathbf{e}_k$ is the edge's attribute, r_k is the index of the receiver node, and s_k is the index of the sender node. For example, E might represent the presence of springs between different balls, and their corresponding spring constants.

Algorithm 1 Steps of computation in a full GN block.

function GRAPHNETWORK(E, V, $\mathbf{u}$)
  for $k \in \{1 \dots N^e\}$ **do**
    $\mathbf{e}'_k \leftarrow \phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right)$ ▷ 1. Compute updated edge attributes
  end for
  for $i \in \{1 \dots N^n\}$ **do**
    let $E'_i = \left\{\left(\mathbf{e}'_k, r_k, s_k\right)\right\}_{r_k=i,\, k=1:N^e}$
    $\bar{\mathbf{e}}'_i \leftarrow \rho^{e\to v}\left(E'_i\right)$ ▷ 2. Aggregate edge attributes per node
    $\mathbf{v}'_i \leftarrow \phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right)$ ▷ 3. Compute updated node attributes
  end for
  let $V' = \left\{\mathbf{v}'\right\}_{i=1:N^v}$
  let $E' = \left\{\left(\mathbf{e}'_k, r_k, s_k\right)\right\}_{k=1:N^e}$
  $\bar{\mathbf{e}}' \leftarrow \rho^{e\to u}\left(E'\right)$ ▷ 4. Aggregate edge attributes globally
  $\bar{\mathbf{v}}' \leftarrow \rho^{v\to u}\left(V'\right)$ ▷ 5. Aggregate node attributes globally
  $\mathbf{u}' \leftarrow \phi^u\left(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}\right)$ ▷ 6. Compute updated global attribute
  return $\left(E', V', \mathbf{u}'\right)$
end function

3.2.2 Internal structure of a GN block

A GN block contains three "update" functions, ϕ, and three "aggregation" functions, ρ,

$$\begin{aligned} \mathbf{e}'_k &= \phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right) & \bar{\mathbf{e}}'_i &= \rho^{e\to v}\left(E'_i\right) \\ \mathbf{v}'_i &= \phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right) & \bar{\mathbf{e}}' &= \rho^{e\to u}\left(E'\right) \\ \mathbf{u}' &= \phi^u\left(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}\right) & \bar{\mathbf{v}}' &= \rho^{v\to u}\left(V'\right) \end{aligned} \tag{1}$$

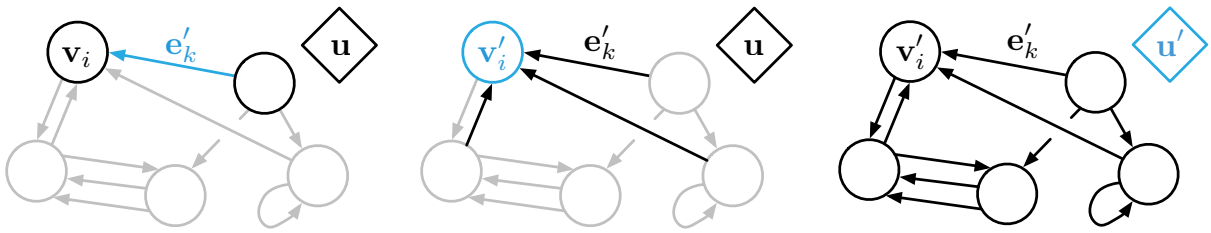
where $E'_i = \left\{\left(\mathbf{e}'_k, r_k, s_k\right)\right\}_{r_k=i,\, k=1:N^e}$, $V' = \left\{\mathbf{v}'_i\right\}_{i=1:N^v}$, and $E' = \bigcup_i E'_i = \left\{\left(\mathbf{e}'_k, r_k, s_k\right)\right\}_{k=1:N^e}$.

The ϕ^e is mapped across all edges to compute per-edge updates, the ϕ^v is mapped across all nodes to compute per-node updates, and the ϕ^u is applied once as the global update. The ρ functions each take a set as input, and reduce it to a single element which represents the aggregated information. Crucially, the ρ functions must be invariant to permutations of their inputs, and should take variable numbers of arguments (e.g., elementwise summation, mean, maximum, etc.).

3.2.3 Computational steps within a GN block

When a graph, G, is provided as input to a GN block, the computations proceed from the edge, to the node, to the global level. Figure 3 shows a depiction of which graph elements are invovled in each of these computations, and Figure 4a shows a full GN block, with its update and aggregation functions. Algorithm 1 shows the following steps of computation:

1. ϕ^e is applied per edge, with arguments $\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right)$, and returns $\mathbf{e}'_k$. In our springs example, this might correspond to the forces or potential energies between two connected balls. The set of resulting per-edge outputs for each node, i, is, $E'_i = \left\{\left(\mathbf{e}'_k, r_k, s_k\right)\right\}_{r_k=i,\, k=1:N^e}$. And $E' = \bigcup_i E'_i = \left\{\left(\mathbf{e}'_k, r_k, s_k\right)\right\}_{k=1:N^e}$ is the set of all per-edge outputs.
2. $\rho^{e\to v}$ is applied to E'_i, and aggregates the edge updates for edges that project to vertex i, into $\bar{\mathbf{e}}'_i$, which will be used in the next step's node update. In our running example, this might correspond to summing all the forces or potential energies acting on the i^{th} ball.



(a) Edge update (b) Node update (c) Global update

Figure 3: Updates in a GN block. Blue indicates the element that is being updated, and black indicates other elements which are involved in the update (note that the pre-update value of the blue element is also used in the update). See Equation 1 for details on the notation.

3. ϕ^v is applied to each node i, to compute an updated node attribute, $\mathbf{v}'_i$. In our running example, ϕ^v may compute something analogous to the updated position, velocity, and kinetic energy of each ball. The set of resulting per-node outputs is, $V' = \{\mathbf{v}'_i\}_{i=1:N^v}$.
4. $\rho^{e\to u}$ is applied to E', and aggregates all edge updates, into $\bar{\mathbf{e}}'$, which will then be used in the next step's global update. In our running example, $\rho^{e\to u}$ may compute the summed forces (which should be zero, in this case, due to Newton's third law) and the springs' potential energies.
5. $\rho^{v\to u}$ is applied to V', and aggregates all node updates, into $\bar{\mathbf{v}}'$, which will then be used in the next step's global update. In our running example, $\rho^{v\to u}$ might compute the total kinetic energy of the system.
6. ϕ^u is applied once per graph, and computes an update for the global attribute, $\mathbf{u}'$. In our running example, ϕ^u might compute something analogous to the net forces and total energy of the physical system.

Note, though we assume this sequence of steps here, the order is not strictly enforced: it is possible to reverse the update functions to proceed from global, to per-node, to per-edge updates, for example. Kearnes et al. (2016) computes edge updates from nodes in a similar manner.

3.2.4 Relational inductive biases in graph networks

Our GN framework imposes several strong relational inductive biases when used as components in a learning process. First, graphs can express arbitrary relationships among entities, which means the GN's input determines how representations interact and are isolated, rather than those choices being determined by the fixed architecture. For example, the assumption that two entities have a relationship—and thus should interact—is expressed by an edge between the entities' corresponding nodes. Similarly, the absence of an edge expresses the assumption that the nodes have no relationship and should not influence each other directly.

Second, graphs represent entities and their relations as sets, which are invariant to permutations. This means GNs are invariant to the order of these elements[7], which is often desirable. For example, the objects in a scene do not have a natural ordering (see Sec. 2.2).

Third, a GN's per-edge and per-node functions are reused across all edges and nodes, respectively. This means GNs automatically support a form of combinatorial generalization (see Section 5.1): because graphs are composed of edges, nodes, and global features, a single GN can operate on graphs of different sizes (numbers of edges and nodes) and shapes (edge connectivity).

[7]Note, an ordering can be imposed by encoding the indices in the node or edge attributes, or via the edges themselves (e.g. by encoding a chain or partial ordering).

4 Design principles for graph network architectures

The GN framework can be used to implement a wide variety of architectures, in accordance with the design principles listed above in Section 3.2, which also correspond to the sub-sections (4.1, 4.2, and 4.3) below. In general, the framework is agnostic to specific attribute representations and functional forms. Here, however, we focus mainly on deep learning architectures, which allow GNs to act as learnable graph-to-graph function approximators.

4.1 Flexible representations

Graph networks support highly flexible graph representations in two ways: first, in terms of the representation of the attributes; and second, in terms of the structure of the graph itself.

4.1.1 Attributes

The global, node, and edge attributes of a GN block can use arbitrary representational formats. In deep learning implementations, real-valued vectors and tensors are most common. However, other data structures such as sequences, sets, or even graphs could also be used.

The requirements of the problem will often determine what representations should be used for the attributes. For example, when the input data is an image, the attributes might be represented as tensors of image patches; however, when the input data is a text document, the attributes might be sequences of words corresponding to sentences.

For each GN block within a broader architecture, the edge and node outputs typically correspond to lists of vectors or tensors, one per edge or node, and the global outputs correspond to a single vector or tensor. This allows a GN's output to be passed to other deep learning building blocks such as MLPs, CNNs, and RNNs.

The output of a GN block can also be tailored to the demands of the task. In particular,

- An *edge-focused* GN uses the edges as output, for example to make decisions about interactions among entities (Kipf et al., 2018; Hamrick et al., 2018).
- A *node-focused* GN uses the nodes as output, for example to reason about physical systems (Battaglia et al., 2016; Chang et al., 2017; Wang et al., 2018b; Sanchez-Gonzalez et al., 2018).
- A *graph-focused* GN uses the globals as output, for example to predict the potential energy of a physical system (Battaglia et al., 2016), the properties of a molecule (Gilmer et al., 2017), or answers to questions about a visual scene (Santoro et al., 2017).

The nodes, edges, and global outputs can also be mixed-and-matched depending on the task. For example, Hamrick et al. (2018) used both the output edge and global attributes to compute a policy over actions.

4.1.2 Graph structure

When defining how the input data will be represented as a graph, there are generally two scenarios: first, the input explicitly specifies the relational structure; and second, the relational structure must be inferred or assumed. These are not hard distinctions, but extremes along a continuum.

Examples of data with more explicitly specified entities and relations include knowledge graphs, social networks, parse trees, optimization problems, chemical graphs, road networks, and physical systems with known interactions. Figures 2a-d illustrate how such data can be expressed as graphs.

Examples of data where the relational structure is not made explicit, and must be inferred or assumed, include visual scenes, text corpora, programming language source code, and multi-agent

systems. In these types of settings, the data may be formatted as a set of entities without relations, or even just a vector or tensor (e.g., an image). If the entities are not specified explicitly, they might be assumed, for instance, by treating each word in a sentence (Vaswani et al., 2017) or each local feature vector in a CNN's output feature map, as a node (Watters et al., 2017; Santoro et al., 2017; Wang et al., 2018c) (Figures 2e-f). Or, it might be possible to use a separate learned mechanism to infer entities from an unstructured signal (Luong et al., 2015; Mnih et al., 2014; Eslami et al., 2016; van Steenkiste et al., 2018). If relations are not available, the simplest approach is to instantiate all possible directed edges between entities (Figure 2f). This can be prohibitive for large numbers of entities, however, because the number of possible edges grows quadratically with the number of nodes. Thus developing more sophisticated ways of inferring sparse structure from unstructured data (Kipf et al., 2018) is an important future direction.

4.2 Configurable within-block structure

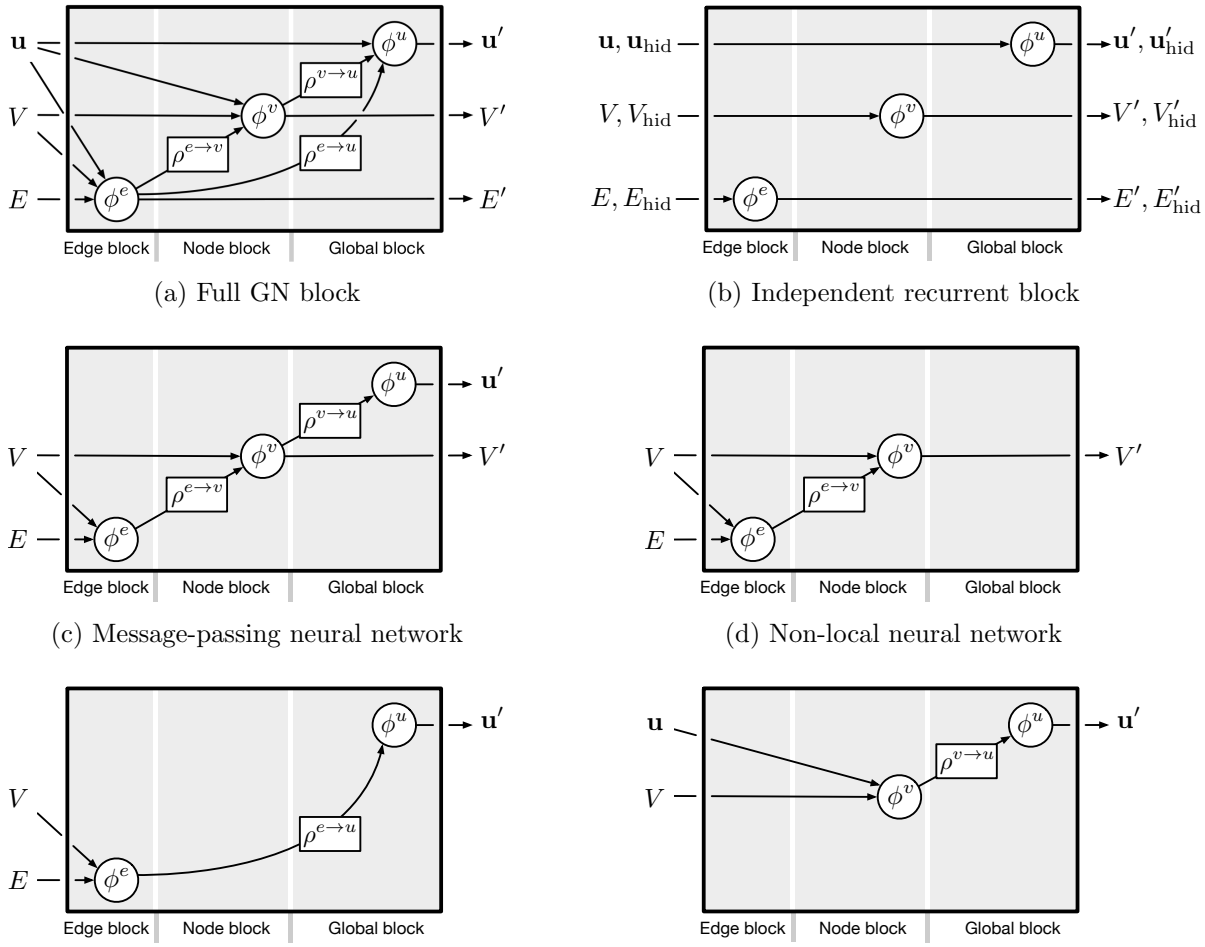
The structure and functions within a GN block can be configured in different ways, which offers flexibility in what information is made available as inputs to its functions, as well as how output edge, node, and global updates are produced. In particular, each ϕ in Equation 1 must be implemented with some function, f, where f's argument signature determines what information it requires as input; in Figure 4, the incoming arrows to each ϕ depict whether $\mathbf{u}$, V, and E are taken as inputs. Hamrick et al. (2018) and Sanchez-Gonzalez et al. (2018) used the full GN block shown in Figure 4a. Their ϕ implementations used neural networks (denoted NN_e, NN_v, and NN_u below, to indicate that they are different functions with different parameters). Their ρ implementations used elementwise summation, but averages and max/min could also be used,

$$
\begin{aligned}
\phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right) &\coloneqq f^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right) &&= \mathrm{NN}_e\left(\left[\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right]\right) \\
\phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right) &\coloneqq f^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right) &&= \mathrm{NN}_v\left(\left[\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right]\right) \\
\phi^u\left(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}\right) &\coloneqq f^u\left(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}\right) &&= \mathrm{NN}_u\left(\left[\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}\right]\right) \\
\rho^{e\to v}\left(E'_i\right) &\coloneqq &&= \sum_{\{k:\, r_k=i\}} \mathbf{e}'_k \\
\rho^{v\to u}\left(V'\right) &\coloneqq &&= \sum_i \mathbf{v}'_i \\
\rho^{e\to u}\left(E'\right) &\coloneqq &&= \sum_k \mathbf{e}'_k
\end{aligned}
\tag{2}
$$

where $[\mathbf{x}, \mathbf{y}, \mathbf{z}]$ indicates vector/tensor concatenation. For vector attributes, a MLP is often used for ϕ, while for tensors such as image feature maps, CNNs may be more suitable.

The ϕ functions can also use RNNs, which requires an additional hidden state as input and output. Figure 4b shows a very simple version of a GN block with RNNs as ϕ functions: there is no message-passing in this formulation, and this type of block might be used for recurrent smoothing of some dynamic graph states. Of course, RNNs as ϕ functions could also be used in a full GN block (Figure 4a).

A variety of other architectures can be expressed in the GN framework, often as different function choices and within-block configurations. The remaining sub-sections explore how a GN's within-block structure can be configured in different ways, with examples of published work which uses such configurations. See the Appendix for details.



(a) Full GN block

(b) Independent recurrent block

(c) Message-passing neural network

(d) Non-local neural network

(e) Relation network

(f) Deep set

Figure 4: Different internal GN block configurations. See Section 3.2 for details on the notation, and Section 4 for details about each variant. (a) A full GN predicts node, edge, and global output attributes based on incoming node, edge, and global attributes. (b) An independent, recurrent update block takes input and hidden graphs, and the ϕ functions are RNNs (Sanchez-Gonzalez et al., 2018). (c) An MPNN (Gilmer et al., 2017) predicts node, edge, and global output attributes based on incoming node, edge, and global attributes. Note that the global prediction does not include aggregated edges. (d) A NLNN (Wang et al., 2018c) only predicts node output attributes. (e) A relation network (Raposo et al., 2017; Santoro et al., 2017) only uses the edge predictions to predict global attributes. (f) A Deep Set (Zaheer et al., 2017) bypasses the edge update and predicts updated global attributes.

4.2.1 Message-passing neural network (MPNN)

Gilmer et al. (2017)'s MPNN generalizes a number of previous architectures and can be translated naturally into the GN formalism. Following the MPNN paper's terminology (see Gilmer et al. (2017), pages 2-4):

- the message function, M_t, plays the role of the GN's ϕ^e, but does not take $\mathbf{u}$ as input,
- elementwise summation is used for the GN's $\rho^{e\to v}$,
- the update function, U_t, plays the role of the GN's ϕ^v,

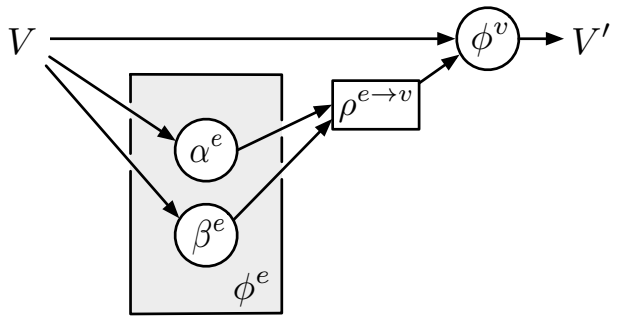


Figure 5: NLNNs as GNs. A schematic showing how NLNNs (Wang et al., 2018c) are implemented by the ϕ^e and $\rho^{e\to v}$ under the GN framework. Typically, NLNNs assume that different regions of an image (or words in a sentence) correspond to nodes in a fully connected graph, and the attention mechanism defines a weighted sum over nodes during the aggregation step.

- the readout function, R, plays the role of the GN's ϕ^u, but does not take $\mathbf{u}$ or E' as input, and thus an analog to the GN's $\rho^{e\to u}$ is not required;
- d_{master} serves a roughly similar purpose to the GN's $\mathbf{u}$, but is defined as an extra node connected to all others, and thus does not influence the edge and global updates directly. It can then be represented in the GN's V.

Figure 4c shows how an MPNN is structured, according to the GN framework. For details and various MPNN architectures, see the Appendix.

4.2.2 Non-local neural networks (NLNN)

Wang et al. (2018c)'s NLNN, which unifies various "intra-/self-/vertex-/graph-attention" approaches (Lin et al., 2017; Vaswani et al., 2017; Hoshen, 2017; Veličković et al., 2018; Shaw et al., 2018), can also be translated into the GN formalism. The label "attention" refers to how the nodes are updated: each node update is based on a weighted sum of (some function of) the node attributes of its neighbors, where the weight between a node and one of its neighbors is computed by a scalar pairwise function between their attributes (and then normalized across neighbors). The published NLNN formalism does not explicitly include edges, and instead computes pairwise attention weights between all nodes. But various NLNN-compliant models, such as the vertex attention interaction network (Hoshen, 2017) and graph attention network (Veličković et al., 2018), are able to handle explicit edges by effectively setting to zero the weights between nodes which do not share an edge.

As Figures 4d and 5 illustrate, the ϕ^e is factored into the scalar pairwise-interaction function which returns the unnormalized attention term, denoted $\alpha^e\left(\mathbf{v}_{r_k}, \mathbf{v}_{s_k}\right) = a'_k$, and a vector-valued non-pairwise term, denoted $\beta^e\left(\mathbf{v}_{s_k}\right) = \mathbf{b}'_k$. In the $\rho^{e\to v}$ aggregation, the a'_k terms are normalized across each receiver's edges, $\mathbf{b}'_k$, and elementwise summed:

$$\begin{aligned}
\phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right) &\coloneqq f^e\left(\mathbf{v}_{r_k}, \mathbf{v}_{s_k}\right) && = \left(\alpha^e\left(\mathbf{v}_{r_k}, \mathbf{v}_{s_k}\right),\ \beta^e\left(\mathbf{v}_{s_k}\right)\right) = \left(a'_k, \mathbf{b}'_k\right) = \mathbf{e}'_k \\
\phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right) &\coloneqq f^v(\bar{\mathbf{e}}'_i) \\
\rho^{e\to v}\left(E'_i\right) &\coloneqq \frac{1}{\sum_{\{k:\, r_k=i\}} a'_k} \sum_{\{k:\, r_k=i\}} a'_k \mathbf{b}'_k
\end{aligned}$$

In the NLNN paper's terminology (see Wang et al. (2018c), pages 2-4):

- their f plays the role of the above α,

- their g plays the role of the above β.

This formulation may be helpful for focusing only on those interactions which are most relevant for the downstream task, especially when the input entities were a set, from which a graph was formed by adding all possible edges between them.

Vaswani et al. (2017)'s multi-headed self-attention mechanism adds an interesting feature, where the ϕ^e and $\rho^{e\to v}$ are implemented by a parallel set of functions, whose results are concatenated together as the final step of $\rho^{e\to v}$. This can be interpreted as using typed edges, where the different types index into different ϕ^e component functions, analogous to Li et al. (2016).

For details and various NLNN architectures, see the Appendix.

4.2.3 Other graph network variants

The full GN (Equation 2) can be used to predict a full graph, or any subset of $(\mathbf{u}', V', E')$, as outlined in Section 4.1.1. For example, to predict a global property of a graph, V' and E' can just be ignored. Similarly, if global, node, or edge attributes are unspecified in the inputs, those vectors can be zero-length, i.e., not taken as explicit input arguments. The same idea applies for other GN variants which do not use the full set of mapping (ϕ) and reduction (ρ) functions. For instance, Interaction Networks (Battaglia et al., 2016; Watters et al., 2017) and the Neural Physics Engine (Chang et al., 2017) use a full GN but for the absence of the global to update the edge properties (see Appendix for details).

Various models, including CommNet (Sukhbaatar et al., 2016), structure2vec (Dai et al., 2016) (in the version of (Dai et al., 2017)), and Gated Graph Sequence Neural Networks (Li et al., 2016) have used a ϕ^e which does not directly compute pairwise interactions, but instead ignore the receiver node, operating only on the sender node and in some cases an edge attribute. This can be expressed by implementations of ϕ^e with the following signatures, such as:

$$\begin{aligned} & \phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right) \coloneqq f^e\left(\mathbf{v}_{s_k}\right) \\ \text{or} \quad & \phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right) \coloneqq \mathbf{v}_{s_k} + f^e\left(\mathbf{e}_k\right) \\ \text{or} \quad & \phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right) \coloneqq f^e\left(\mathbf{e}_k, \mathbf{v}_{s_k}\right). \end{aligned}$$

See the Appendix for further details.

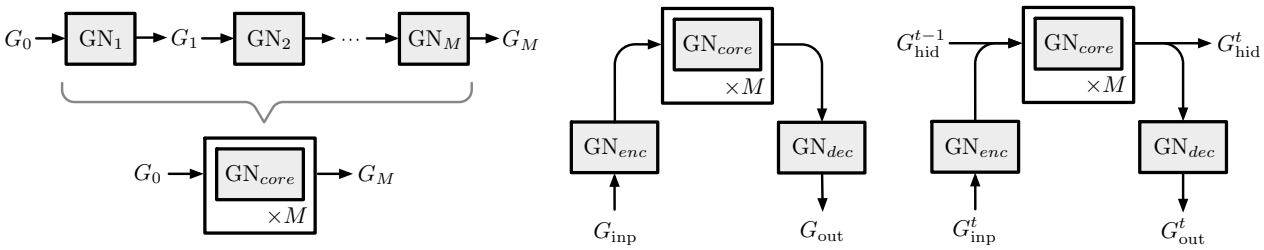
Relation Networks (Raposo et al., 2017; Santoro et al., 2017) bypass the node update entirely and predict the global output from pooled edge information directly (see also Figure 4e),

$$\begin{aligned} \phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right) \coloneqq f^e\left(\mathbf{v}_{r_k}, \mathbf{v}_{s_k}\right) &= \mathrm{NN}_e\left(\left[\mathbf{v}_{r_k}, \mathbf{v}_{s_k}\right]\right) \\ \phi^u\left(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}\right) \coloneqq f^u\left(\bar{\mathbf{e}}'\right) &= \mathrm{NN}_u\left(\bar{\mathbf{e}}'\right) \\ \rho^{e\to u}\left(E'\right) \coloneqq &= \sum_k \mathbf{e}'_k \end{aligned}$$

Deep Sets (Zaheer et al., 2017) bypass the edges update completely and predict the global output from pooled nodes information directly (Figure 4f),

$$\begin{aligned} \phi^v\left(\bar{\mathbf{e}}_i, \mathbf{v}_i, \mathbf{u}\right) \coloneqq f^v\left(\mathbf{v}_i, \mathbf{u}\right) &= \mathrm{NN}_v\left(\left[\mathbf{v}_i, \mathbf{u}\right]\right) \\ \phi^u\left(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}\right) \coloneqq f^u\left(\bar{\mathbf{v}}'\right) &= \mathrm{NN}_u\left(\bar{\mathbf{v}}'\right) \\ \rho^{v\to u}\left(V'\right) \coloneqq &= \sum_i \mathbf{v}'_i \end{aligned}$$

PointNet (Qi et al., 2017) use similar update rule, with a max-aggregation for $\rho^{v\to u}$ and a two-step node update.



(a) Composition of GN blocks (b) Encode-process-decode (c) Recurrent GN architecture

Figure 6: (a) An example composing multiple GN blocks in sequence to form a GN "core". Here, the GN blocks can use shared weights, or they could be independent. (b) The *encode-process-decode* architecture, which is a common choice for composing GN blocks (see Section 4.3). Here, a GN encodes an input graph, which is then processed by a GN core. The output of the core is decoded by a third GN block into an output graph, whose nodes, edges, and/or global attributes would be used for task-specific purposes. (c) The encode-process-decode architecture applied in a sequential setting in which the core is also unrolled over time (potentially using a GRU or LSTM architecture), in addition to being repeated within each time step. Here, merged lines indicate concatenation, and split lines indicate copying.

4.3 Composable multi-block architectures

A key design principle of graph networks is constructing complex architectures by composing GN blocks. We defined a GN block as always taking a graph comprised of edge, node, and global elements as input, and returning a graph with the same constituent elements as output (simply passing through the input elements to the output when those elements are not explicitly updated). This graph-to-graph input/output interface ensures that the output of one GN block can be passed as input to another, even if their internal configurations are different, similar to the tensor-to-tensor interface of the standard deep learning toolkit. In the most basic form, two GN blocks, GN_1 and GN_2, can be composed as $\mathrm{GN}_1 \circ \mathrm{GN}_2$ by passing the output of the first as input to the second: $G' = \mathrm{GN}_2(\mathrm{GN}_1(G))$.

Arbitrary numbers of GN blocks can be composed, as show in Figure 6a. The blocks can be unshared (different functions and/or parameters, analogous to layers of a CNN), $\mathrm{GN}_1 \neq \mathrm{GN}_2 \neq \cdots \neq \mathrm{GN}_M$, or shared (reused functions and parameters, analogous to an unrolled RNN), $\mathrm{GN}_1 = \mathrm{GN}_2 = \cdots = \mathrm{GN}_M$. The white box around the GN_{core} in Figure 6a represents M repeated internal processing sub-steps, with either shared or unshared GN blocks. Shared configurations are analogous to message-passing (Gilmer et al., 2017), where the same local update procedure is applied iteratively to propagate information across the structure (Figure 7). If we exclude the global $\mathbf{u}$ (which aggregates information from across the nodes and edges), the information that a node has access to after m steps of propagation is determined by the set of nodes and edges that are at most m hops away. This can be interpreted as breaking down a complex computation into smaller elementary steps. The steps can also be used to capture sequentiality in time. In our ball-spring example, if each propagation step predicts the physical dynamics over one time step of duration Δt, then the M propagation steps result in a total simulation time of, $M \cdot \Delta t$.

A common architecture design is what we call the *encode-process-decode* configuration (Hamrick et al. (2018); also see Figure 6ba): an input graph, G_{inp} is transformed into a latent representation, G_0, by an encoder, $\mathrm{GN}_{\mathrm{enc}}$; a shared core block, $\mathrm{GN}_{\mathrm{core}}$, is applied M times to return G_M; and finally an output graph, G_{out}, is decoded by $\mathrm{GN}_{\mathrm{dec}}$. For example, in our running example, the encoder might compute the initial forces and interaction energies between the balls, the core might

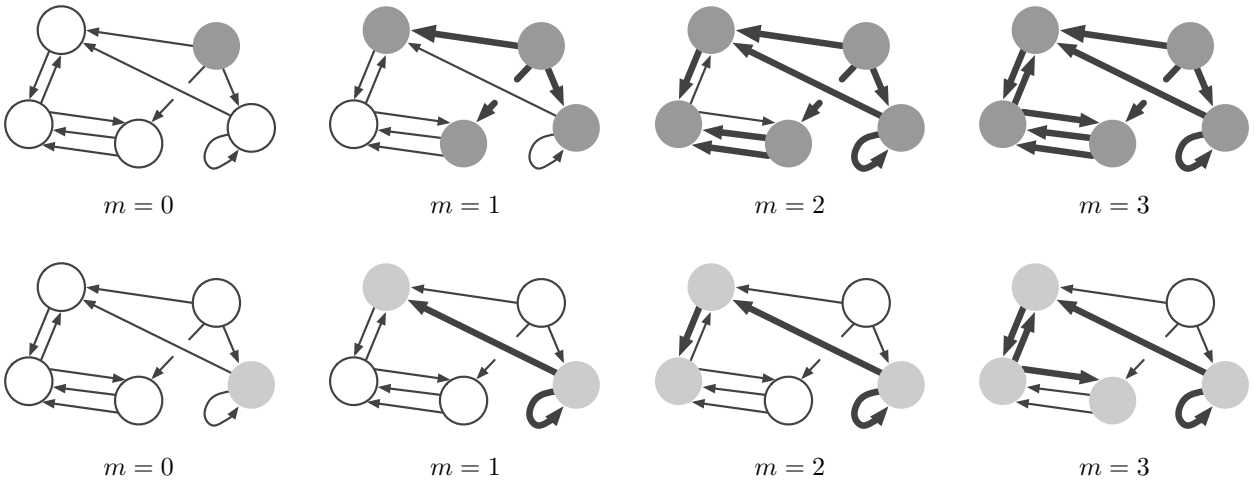


Figure 7: Example of message passing. Each row highlights the information that diffuses through the graph starting from a particular node. In the top row, the node of interest is in the upper right; in the bottom row, the node of interest is in the bottom right. Shaded nodes indicate how far information from the original node can travel in m steps of message passing; bolded edges indicate which edges that information has the potential to travel across. Note that during the full message passing procedure, this propagation of information happens simultaneously for all nodes and edges in the graph (not just the two shown here).

apply an elementary dynamics update, and the decoder might read out the final positions from the updated graph state.

Similar to the encode-process-decode design, recurrent GN-based architectures can be built by maintaining a hidden graph, G^t_{hid}, taking as input an observed graph, G^t_{inp}, and returning an output graph, G^t_{out}, on each step (see Figure 6c). This type of architecture can be particularly useful for predicting sequences of graphs, such as predicting the trajectory of a dynamical system over time (e.g. Sanchez-Gonzalez et al., 2018). The encoded graph, output by GN_{enc}, must have the same structure as G^t_{hid}, and they can be easily combined by concatenating their corresponding $\mathbf{e}_k$, $\mathbf{v}_i$, and $\mathbf{u}$ vectors (where the upward arrow merges into the left-hand horizontal arrow in Figure 6c), before being passed to GN_{core}. For the output, the G^t_{hid} is copied (where the right-hand horizontal arrow splits into the downward arrow in Figure 6c) and decoded by GN_{dec}. This design reuses GN blocks in several ways: GN_{enc}, GN_{dec}, and GN_{core} are shared across each step, t; and within each step, GN_{core} may perform multiple shared sub-steps.

Various other techniques for designing GN-based architectures can be useful. Graph skip connections, for example, would concatenate a GN block's input graph, G_m, with its output graph, G_{m+1}, before proceeding to further computations. Merging and smoothing input and hidden graph information, as in Figure 6c, can use LSTM- or GRU-style gating schemes, instead of simple concatenation (Li et al., 2016). Or distinct, recurrent GN blocks (e.g. Figure 4b) can be composed before and/or after other GN blocks, to improve stability in the representations over multiple propagation steps (Sanchez-Gonzalez et al., 2018).

4.4 Implementing graph networks in code

Similar to CNNs (see Figure 1), which are naturally parallelizable (e.g. on GPUs), GNs have a natural parallel structure: since the ϕ^e and ϕ^v functions in Equation 1 are shared over the edges and nodes, respectively, they can be computed in parallel. In practice, this means that with respect

to ϕ^e and ϕ^v, the nodes and edges can be treated like the batch dimension in typical mini-batch training regimes. Moreover, several graphs can be naturally batched together by treating them as disjoint components of a larger graph. With some additional bookkeeping, this allows batching together the computations made on several independent graphs.

Reusing ϕ^e and ϕ^v also improves GNs' sample efficiency. Again, analogous to a convolutional kernel, the number of samples which are used to optimize a GN's ϕ^e and ϕ^v functions is the number of edges and nodes, respectively, across all training graphs. For example, in the balls example from Sec. 3.2, a scene with four balls which are all connected by springs will provide twelve (4×3) examples of the contact interaction between them.

4.5 Summary

In this section, we have discussed the design principles behind graph networks: flexible representations, configurable within-block structure, and composable multi-block architectures. These three design principles combine in our framework which is extremely flexible and applicable to a wide range of domains ranging from perception, language, and symbolic reasoning. And, as we will see in the remainder of this paper, the strong relational inductive bias possessed by graph networks supports combinatorial generalization, thus making it a powerful tool both in terms of implementation and theory.

5 Discussion

In this paper, we analyzed the extent to which relational inductive bias exists in deep learning architectures like MLPs, CNNs, and RNNs, and concluded that while CNNs and RNNs do contain relational inductive biases, they cannot naturally handle more structured representations such as sets or graphs. We advocated for building stronger relational inductive biases into deep learning architectures by highlighting an underused deep learning building block called a *graph network*, which performs computations over graph-structured data. Our graph network framework unifies existing approaches that also operate over graphs, and provides a straightforward interface for assembling graph networks into complex, sophisticated architectures.

5.1 Combinatorial generalization in graph networks

The structure of GNs naturally supports combinatorial generalization because they do not perform computations strictly at the system level, but also apply shared computations across the entities and across the relations as well. This allows never-before-seen systems to be reasoned about, because they are built from familiar components, in a way that reflects von Humboldt's "infinite use of finite means" (Humboldt, 1836; Chomsky, 1965).

A number of studies have explored GNs' capacity for combinatorial generalization. Battaglia et al. (2016) found that GNs trained to make one-step physical state predictions could simulate thousands of future time steps, and also exhibit accurate zero-shot transfer to physical systems with double, or half, the number of entities experienced during training. Sanchez-Gonzalez et al. (2018) found similar results in more complex physical control settings, including that GNs trained as forward models on simulated multi-joint agents could generalize to agents with new numbers of joints. Hamrick et al. (2018) and Wang et al. (2018b) each found that GN-based decision-making policies could transfer to novel numbers of entities as well. In combinatorial optimization problems, Bello et al. (2016); Nowak et al. (2017); Dai et al. (2017); Kool and Welling (2018) showed that GNs could generalize well to problems of much different sizes than they had been trained on. Similarly, Toyer

et al. (2017) showed generalization to different sizes of planning problems, and Hamilton et al. (2017) showed generalization to producing useful node embeddings for previously unseen data. On boolean SAT problems, Selsam et al. (2018) demonstrated generalization both to different problem sizes and across problem distributions: their model retained good performance upon strongly modifying the distribution of the input graphs and its typical local structure.

These striking examples of combinatorial generalization are not entirely surprising, given GNs' entity- and relation-centric organization, but nonetheless provide important support for the view that embracing explicit structure and flexible learning is a viable approach toward realizing better sample efficiency and generalization in modern AI.

5.2 Limitations of graph networks

One limitation of GNs' and MPNNs' form of learned message-passing (Shervashidze et al., 2011) is that it cannot be guaranteed to solve some classes of problems, such as discriminating between certain non-isomorphic graphs. Kondor et al. (2018) suggested that covariance[8] (Cohen and Welling, 2016; Kondor and Trivedi, 2018), rather than invariance to permutations of the nodes and edges is preferable, and proposed "covariant compositional networks" which can preserve structural information, and allow it to be ignored only if desired.

More generally, while graphs are a powerful way of representing structure information, they have limits. For example, notions like recursion, control flow, and conditional iteration are not straightforward to represent with graphs, and, minimally, require additional assumptions (e.g., in interpreting abstract syntax trees). Programs and more "computer-like" processing can offer greater representational and computational expressivity with respect to these notions, and some have argued they are an important component of human cognition (Tenenbaum et al., 2011; Lake et al., 2015; Goodman et al., 2015).

5.3 Open questions

Although we are excited about the potential impacts that graph networks can have, we caution that these models are only one step forward. Realizing the full potential of graph networks will likely be far more challenging than organizing their behavior under one framework, and indeed, there are a number of unanswered questions regarding the best ways to use graph networks.

One pressing question is: where do the graphs come from that graph networks operate over? One of the hallmarks of deep learning has been its ability to perform complex computations over raw sensory data, such as images and text, yet it is unclear the best ways to convert sensory data into more structured representations like graphs. One approach (which we have already discussed) assumes a fully connected graph structure between spatial or linguistic entities, such as in the literature on self-attention (Vaswani et al., 2017; Wang et al., 2018c). However, such representations may not correspond exactly to the "true" entities (e.g., convolutional features do not directly correspond to objects in a scene). Moreover, many underlying graph structures are much more sparse than a fully connected graph, and it is an open question how to induce this sparsity. Several lines of active research are exploring these issues (Watters et al., 2017; van Steenkiste et al., 2018; Li et al., 2018; Kipf et al., 2018) but as of yet there is no single method which can reliably extract discrete entities from sensory data. Developing such a method is an exciting challenge for future research, and once solved will likely open the door for much more powerful and flexible reasoning algorithms.

[8]Covariance means, roughly, that the activations vary in a predictable way as a function of the ordering of the incoming edges.

A related question is how to adaptively modify graph structures during the course of computation. For example, if an object fractures into multiple pieces, a node representing that object also ought to split into multiple nodes. Similarly, it might be useful to only represent edges between objects that are in contact, thus requiring the ability to add or remove edges depending on context. The question of how to support this type of adaptivity is also actively being researched, and in particular, some of the methods used for identifying the underlying structure of a graph may be applicable (e.g. Li et al., 2018; Kipf et al., 2018).

Human cognition makes the strong assumption that the world is composed of objects and relations (Spelke and Kinzler, 2007), and because GNs make a similar assumption, their behavior tends to be more interpretable. The entities and relations that GNs operate over often correspond to things that humans understand (such as physical objects), thus supporting more interpretable analysis and visualization (e.g., as in Selsam et al., 2018). An interesting direction for future work is to further explore the interpretability of the behavior of graph networks.

5.4 Integrative approaches for learning and structure

While our focus here has been on graphs, one takeaway from this paper is less about graphs themselves and more about the approach of blending powerful deep learning approaches with structured representations. We are excited by related approaches which have explored this idea for other types of structured representations and computations, such as linguistic trees (Socher et al., 2011a,b, 2012, 2013; Tai et al., 2015; Andreas et al., 2016), partial tree traversals in a state-action graph (Guez et al., 2018; Farquhar et al., 2018), hierarchical action policies (Andreas et al., 2017), "capsules" (Sabour et al., 2017), and programs (Parisotto et al., 2017). Other methods have attempted to capture different types of structure by mimicking key hardware and software components in computers and how they transfer information between each other, such as persistent slotted storage, registers, memory I/O controllers, stacks, and queues (e.g. Dyer et al., 2015; Grefenstette et al., 2015; Joulin and Mikolov, 2015; Sukhbaatar et al., 2015; Kurach et al., 2016; Graves et al., 2016).

5.5 Conclusion

Recent advances in AI, propelled by deep learning, have been transformative across many important domains. Despite this, a vast gap between human and machine intelligence remains, especially with respect to efficient, generalizable learning. We argue for making combinatorial generalization a top priority for AI, and advocate for embracing integrative approaches which draw on ideas from human cognition, traditional computer science, standard engineering practice, and modern deep learning. Here we explored flexible learning-based approaches which implement strong relational inductive biases to capitalize on explicitly structured representations and computations, and presented a framework called *graph networks*, which generalize and extend various recent approaches for neural networks applied to graphs. Graph networks are designed to promote building complex architectures using customizable graph-to-graph building blocks, and their relational inductive biases promote combinatorial generalization and improved sample efficiency over other standard machine learning building blocks.

Despite their benefits and potential, however, learnable models which operate on graphs are only a stepping stone on the path toward human-like intelligence. We are optimistic about a number of other relevant, and perhaps underappreciated, research directions, including marrying learning-based approaches with programs (Ritchie et al., 2016; Andreas et al., 2016; Gaunt et al., 2016; Evans and Grefenstette, 2018; Evans et al., 2018), developing model-based approaches with an emphasis on abstraction (Kansky et al., 2017; Konidaris et al., 2018; Zhang et al., 2018; Hay et al.,

2018), investing more heavily in meta-learning (Wang et al., 2016, 2018a; Finn et al., 2017), and exploring multi-agent learning and interaction as a key catalyst for advanced intelligence (Nowak, 2006; Ohtsuki et al., 2006). These directions each involve rich notions of entities, relations, and combinatorial generalization, and can potentially benefit, and benefit from, greater interaction with approaches for learning relational reasoning over explicitly structured representations.

Acknowledgements

We thank Tobias Pfaff, Danilo Rezende, Nando de Freitas, Murray Shanahan, Thore Graepel, John Jumper, Demis Hassabis, and the broader DeepMind and Google communities for valuable feedback and support.

References

Allamanis, M., Brockschmidt, M., and Khademi, M. (2018). Learning to represent programs with graphs. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Allamanis, M., Chanthirasegaran, P., Kohli, P., and Sutton, C. (2017). Learning continuous semantic representations of symbolic expressions. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Anderson, J. R. (1982). Acquisition of cognitive skill. *Psychological Review*, 89(4):369.

Andreas, J., Klein, D., and Levine, S. (2017). Modular multitask reinforcement learning with policy sketches. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Andreas, J., Rohrbach, M., Darrell, T., and Klein, D. (2016). Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 39–48.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Battaglia, P., Pascanu, R., Lai, M., Rezende, D. J., et al. (2016). Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems*, pages 4502–4510.

Battaglia, P. W., Hamrick, J. B., and Tenenbaum, J. B. (2013). Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, 110(45):18327–18332.

Bello, I., Pham, H., Le, Q. V., Norouzi, M., and Bengio, S. (2016). Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*.

Bobrow, D. G. and Hinton, G. E., editors (1990). *Artificial Intelligence*, volume 46. Elsevier Science Publishers Ltd., Essex, UK. Special Issue 1-2: On Connectionist Symbol Processing.

Bojchevski, A., Shchur, O., Zügner, D., and Günnemann, S. (2018). Netgan: Generating graphs via random walks. *arXiv preprint arXiv:1803.00816*.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795.

Botvinick, M. M. (2008). Hierarchical models of behavior and prefrontal function. *Trends in Cognitive Sciences*, 12(5):201–208.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. (2017). Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. (2014). Spectral networks and locally connected networks on graphs. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Chang, M. B., Ullman, T., Torralba, A., and Tenenbaum, J. B. (2017). A compositional object-based approach to learning physical dynamics. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Chen, X., Li, L., Fei-Fei, L., and Gupta, A. (2018a). Iterative visual reasoning beyond convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Chen, X., Liu, C., and Song, D. (2018b). Tree-to-tree neural networks for program translation. In *Workshops of the International Conference on Learning Representations (ICLR)*.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceeding of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Chomsky, N. (1957). *Syntactic Structures*. Mouton & Co.

Chomsky, N. (1965). *Aspects of the Theory of Syntax*. MIT Press.

Cohen, T. and Welling, M. (2016). Group equivariant convolutional networks. In *International Conference on Machine Learning*, pages 2990–2999.

Craik, K. J. W. (1943). *The Nature of Explanation*. Cambridge University Press.

Cui, Z., Henrickson, K., Ke, R., and Wang, Y. (2018). High-order graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. *arXiv preprint arXiv:1802.07007*.

Dai, H., Dai, B., and Song, L. (2016). Discriminative embeddings of latent variable models for structured data. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Dai, H., Khalil, E. B., Zhang, Y., Dilkina, B., and Song, L. (2017). Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*.

De Cao, N. and Kipf, T. (2018). MolGAN: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*.

Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852.

Denil, M., Colmenarejo, S. G., Cabi, S., Saxton, D., and de Freitas, N. (2017). Programmable agents. *arXiv preprint arXiv:1706.06383*.

Devlin, J., Uesato, J., Singh, R., and Kohli, P. (2017). Semantic code repair using neuro-symbolic transformation networks. *arXiv preprint arXiv:1710.11054*.

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. (2015). Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems*, pages 2224–2232.

Dyer, C., Ballesteros, M., Ling, W., Matthews, A., and Smith, N. A. (2015). Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Džeroski, S., De Raedt, L., and Driessens, K. (2001). Relational reinforcement learning. *Machine Learning*, 43(1-2):7–52.

Eliasmith, C. (2013). *How to build a brain: A neural architecture for biological cognition*. Oxford University Press.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2):179–211.

Elman, J. L. (1991). Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, 7(2-3):195–225.

Eslami, S. A., Heess, N., Weber, T., Tassa, Y., Szepesvari, D., Hinton, G. E., et al. (2016). Attend, infer, repeat: Fast scene understanding with generative models. In *Advances in Neural Information Processing Systems*, pages 3225–3233.

Evans, R. and Grefenstette, E. (2018). Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research*, 61:1–64.

Evans, R., Saxton, D., Amos, D., Kohli, P., and Grefenstette, E. (2018). Can neural networks understand logical entailment? In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Farquhar, G., Rocktäschel, T., Igl, M., and Whiteson, S. (2018). TreeQN and ATreeC: Differentiable tree planning for deep reinforcement learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*.

Fodor, J. A. (1975). *The Language of Thought*. Harvard University Press.

Fodor, J. A. and Pylyshyn, Z. W. (1988). Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202.

Garcia, V. and Bruna, J. (2018). Few-shot learning with graph neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

García-Durán, A. and Niepert, M. (2017). Learning graph representations with embedding propagation. *arXiv preprint arXiv:1710.03059*.

Garnelo, M., Arulkumaran, K., and Shanahan, M. (2016). Towards deep symbolic reinforcement learning. *arXiv preprint arXiv:1609.05518*.

Gaunt, A. L., Brockschmidt, M., Kushman, N., and Tarlow, D. (2016). Differentiable programs with neural libraries. *arXiv preprint arXiv:1611.02109*.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58.

Gentner, D. and Markman, A. B. (1997). Structure mapping in analogy and similarity. *American Psychologist*, 52(1):45.

Getoor, L. and Taskar, B. (2007). *Introduction to Statistical Relational Learning*. MIT press.

Ghahramani, Z. (2015). Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep Learning*. MIT Press.

Goodman, N. (1955). The new riddle of induction. In *Fact, Fiction, and Forecast*, pages 59–83. Harvard University Press.

Goodman, N., Mansinghka, V., Roy, D. M., Bonawitz, K., and Tenenbaum, J. B. (2012). Church: a language for generative models. *arXiv preprint arXiv:1206.3255*.

Goodman, N. D., Tenenbaum, J. B., and Gerstenberg, T. (2015). Concepts in a probabilistic language of thought. In Margolis, E. and Laurence, S., editors, *The Conceptual Mind: New Directions in the Study of Concepts*. MIT Press.

Goodwin, G. P. and Johnson-Laird, P. (2005). Reasoning about relations. *Psychological Review*, 112(2):468.

Gori, M., Monfardini, G., and Scarselli, F. (2005). A new model for learning in graph domains. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 2, pages 729–734. IEEE.

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S. G., Grefenstette, E., Ramalho, T., Agapiou, J., et al. (2016). Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471.

Grefenstette, E., Hermann, K. M., Suleyman, M., and Blunsom, P. (2015). Learning to transduce with unbounded memory. In *Advances in Neural Information Processing Systems*, pages 1828–1836.

Griffiths, T. L., Chater, N., Kemp, C., Perfors, A., and Tenenbaum, J. B. (2010). Probabilistic models of cognition: Exploring representations and inductive biases. *Trends in Cognitive Sciences*, 14(8):357–364.

Grover, A. and Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM.

Guez, A., Weber, T., Antonoglou, I., Simonyan, K., Vinyals, O., Wierstra, D., Munos, R., and Silver, D. (2018). Learning to search with MCTSnets. *arXiv preprint arXiv:1802.04697*.

Gulcehre, C., Denil, M., Malinowski, M., Razavi, A., Pascanu, R., Hermann, K. M., Battaglia, P., Bapst, V., Raposo, D., Santoro, A., and de Freitas, N. (2018). Hyperbolic attention networks. *arXiv preprint arXiv:1805.09786*.

Hamaguchi, T., Oiwa, H., Shimbo, M., and Matsumoto, Y. (2017). Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1025–1035.

Hamrick, J., Allen, K., Bapst, V., Zhu, T., McKee, K., Tenenbaum, J., and Battaglia, P. (2018). Relational inductive bias for physical construction in humans and machines. In *Proceedings of the 40th Annual Conference of the Cognitive Science Society*.

Hamrick, J. B., Ballard, A. J., Pascanu, R., Vinyals, O., Heess, N., and Battaglia, P. W. (2017). Metacontrol for adaptive imagination-based optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Hay, N., Stark, M., Schlegel, A., Wendelken, C., Park, D., Purdy, E., Silver, T., Phoenix, D. S., and George, D. (2018). Behavior is everything–towards representing concepts with sensorimotor contingencies. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Henaff, M., Bruna, J., and LeCun, Y. (2015). Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*.

Hinton, G. E. (1990). Mapping part-whole hierarchies into connectionist networks. *Artificial Intelligence*, 46(1-2):47–75.

Hjort, N. L., Holmes, C., Müller, P., and Walker, S. G. (2010). *Bayesian Nonparametrics*. Cambridge University Press.

Hoshen, Y. (2017). Vain: Attentional multi-agent predictive modeling. In *Advances in Neural Information Processing Systems*, pages 2698–2708.

Hudson, D. A. and Manning, C. D. (2018). Compositional attention networks for machine reasoning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Humboldt, W. (1999/1836). *On Language: On the diversity of human language construction and its influence on the mental development of the human species*. Cambridge University Press.

Hummel, J. E. and Holyoak, K. J. (2003). A symbolic-connectionist theory of relational inference and generalization. *Psychological Review*, 110(2):220.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning (ICML)*.

Johnson, D. D. (2017). Learning graphical state transitions. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Joulin, A. and Mikolov, T. (2015). Inferring algorithmic patterns with stack-augmented recurrent nets. In *Advances in Neural Information Processing Systems*, pages 190–198.

Kansky, K., Silver, T., Mély, D. A., Eldawy, M., Lázaro-Gredilla, M., Lou, X., Dorfman, N., Sidor, S., Phoenix, S., and George, D. (2017). Schema networks: Zero-shot transfer with a generative causal model of intuitive physics. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Kearnes, S., McCloskey, K., Berndl, M., Pande, V., and Riley, P. (2016). Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608.

Kemp, C. and Tenenbaum, J. B. (2008). The discovery of structural form. *Proceedings of the National Academy of Sciences*, 105(31):10687–10692.

Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., and Zemel, R. (2018). Neural relational inference for interacting systems. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT press.

Kondor, R., Son, H. T., Pan, H., Anderson, B., and Trivedi, S. (2018). Covariant compositional networks for learning graphs. *arXiv preprint arXiv:1801.02144*.

Kondor, R. and Trivedi, S. (2018). On the generalization of equivariance and convolution in neural networks to the action of compact groups. *arXiv preprint arXiv:1802.03690*.

Konidaris, G., Kaelbling, L. P., and Lozano-Perez, T. (2018). From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 61:215–289.

Kool, W. and Welling, M. (2018). Attention solves your TSP. *arXiv preprint arXiv:1803.08475*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105.

Kurach, K., Andrychowicz, M., and Sutskever, I. (2016). Neural random-access machines. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Lake, B. M. and Baroni, M. (2018). Still not systematic after all these years: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. (2017). Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. (2016). Gated graph sequence neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Li, Y., Vinyals, O., Dyer, C., Pascanu, R., and Battaglia, P. (2018). Learning deep generative models of graphs. In *Workshops at the International Conference on Learning Representations (ICLR)*.

Lin, Z., Feng, M., Santos, C. N. d., Yu, M., Xiang, B., Zhou, B., and Bengio, Y. (2017). A structured self-attentive sentence embedding. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Liu, H., Simonyan, K., Vinyals, O., Fernando, C., and Kavukcuoglu, K. (2018). Hierarchical representations for efficient architecture search. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Marcus, G. (2001). The algebraic mind.

Marcus, G. (2018a). Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*.

Marcus, G. (2018b). Innateness, alphazero, and artificial intelligence. *arXiv preprint arXiv:1801.05667*.

McClelland, J. L. (1994). The interaction of nature and nurture in development: A parallel distributed processing perspective. *International perspectives on psychological science*, 1:57–88.

McClelland, J. L. and Rumelhart, D. E. (1981). An interactive activation model of context effects in letter perception: I. an account of basic findings. *Psychological Review*, 88(5):375.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

Mitchell, T. M. (1980). *The need for biases in learning generalizations*. Department of Computer Science, Laboratory for Computer Science Research, Rutgers Univ. New Jersey.

Mnih, V., Heess, N., Graves, A., et al. (2014). Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529.

Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., and Bronstein, M. M. (2017). Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Moravčík, M., Schmid, M., Burch, N., Lisỳ, V., Morrill, D., Bard, N., Davis, T., Waugh, K., Johanson, M., and Bowling, M. (2017). Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513.

Narayanan, A., Chandramohan, M., Chen, L., Liu, Y., and Saminathan, S. (2016). subgraph2vec: Learning distributed representations of rooted sub-graphs from large graphs. In *Workshops at the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Liu, Y., and Jaiswal, S. (2017). graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*.

Navon, D. (1977). Forest before trees: The precedence of global features in visual perception. *Cognitive Psychology*, 9(3):353–383.

Niepert, M., Ahmed, M., and Kutzkov, K. (2016). Learning convolutional neural networks for graphs. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 2014–2023.

Nilsson, N. J. and Fikes, R. E. (1970). Strips: A new approach to the application of theorem proving to problem solving. Technical report, SRI International, Menlo Park, CA Artificial Intelligence Center.

Nowak, A., Villar, S., Bandeira, A. S., and Bruna, J. (2017). A note on learning algorithms for quadratic assignment with graph neural networks. In *Proceedings of the Principled Approaches to Deep Learning Workshop (PADL) at the International Conference of Machine Learning (ICML)*.

Nowak, M. A. (2006). Five rules for the evolution of cooperation. *science*, 314(5805):1560–1563.

Ohtsuki, H., Hauert, C., Lieberman, E., and Nowak, M. A. (2006). A simple rule for the evolution of cooperation on graphs and social networks. *Nature*, 441(7092):502.

Oñoro-Rubio, D., Niepert, M., García-Durán, A., González-Sánchez, R., and López-Sastre, R. J. (2017). Representation learning for visual-relational knowledge graphs. *arXiv preprint arXiv:1709.02314*.

Parisotto, E., Mohamed, A.-r., Singh, R., Li, L., Zhou, D., and Kohli, P. (2017). Neuro-symbolic program synthesis. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Pascanu, R., Li, Y., Vinyals, O., Heess, N., Buesing, L., Racanière, S., Reichert, D., Weber, T., Wierstra, D., and Battaglia, P. (2017). Learning model-based planning from scratch. *arXiv preprint arXiv:1707.06170*.

Pearl, J. (1986). Fusion, propagation, and structuring in belief networks. *Artificial intelligence*, 29(3):241–288.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann.

Pearl, J. (2009). *Causality: Models, Reasoning and Inference*. Cambridge University Press, New York, NY, USA, 2nd edition.

Pearl, J. (2018). Theoretical impediments to machine learning with seven sparks from the causal revolution. *arXiv preprint arXiv:1801.04016*.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM.

Pinker, S. and Prince, A. (1988). On language and connectionism: Analysis of a parallel distributed processing model of language acquisition. *Cognition*, 28(1-2):73–193.

Plate, T. A. (1995). Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6(3):623–641.

Plaut, D. C., McClelland, J. L., Seidenberg, M. S., and Patterson, K. (1996). Understanding normal and impaired word reading: computational principles in quasi-regular domains. *Psychological Review*, 103(1):56.

Pollack, J. B. (1990). Recursive distributed representations. *Artificial Intelligence*, 46(1-2):77–105.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Raposo, D., Santoro, A., Barrett, D., Pascanu, R., Lillicrap, T., and Battaglia, P. (2017). Discovering objects and their relations from entangled scene representations. In *Workshops at the International Conference on Learning Representations (ICLR)*.

Reed, S. and De Freitas, N. (2016). Neural programmer-interpreters. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Ritchie, D., Horsfall, P., and Goodman, N. D. (2016). Deep amortized inference for probabilistic programs. *arXiv preprint arXiv:1610.05735*.

Rosenblatt, F. (1961). Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc., Buffalo, NY.

Rumelhart, D. E., McClelland, J. L., Group, P. R., et al. (1987). *Parallel Distributed Processing*, volume 1. MIT Press.

Russell, S. J. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach (3rd Edition)*. Pearson.

Sabour, S., Frosst, N., and Hinton, G. E. (2017). Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pages 3859–3869.

Sanchez-Gonzalez, A., Heess, N., Springenberg, J. T., Merel, J., Riedmiller, M., Hadsell, R., and Battaglia, P. (2018). Graph networks as learnable physics engines for inference and control. In *Proceedings of the 35th International Conference on Machine Learning (ICLR)*.

Santoro, A., Raposo, D., Barrett, D. G., Malinowski, M., Pascanu, R., Battaglia, P., and Lillicrap, T. (2017). A simple neural network module for relational reasoning. In *Advances in Neural Information Processing Systems*.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009a). Computational capabilities of graph neural networks. *IEEE Transactions on Neural Networks*, 20(1):81–102.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009b). The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Scarselli, F., Yong, S. L., Gori, M., Hagenbuchner, M., Tsoi, A. C., and Maggini, M. (2005). Graph neural networks for ranking web pages. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 666–672. IEEE.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

Selsam, D., Lamm, M., Bunz, B., Liang, P., de Moura, L., and Dill, D. L. (2018). Learning a sat solver from single-bit supervision. *arXiv preprint arXiv:1802.03685*.

Shalev-Shwartz, S., Shamir, O., and Shammah, S. (2017). Failures of gradient-based deep learning. *arXiv preprint arXiv:1703.07950*.

Shaw, P., Uszkoreit, J., and Vaswani, A. (2018). Self-attention with relative position representations. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Shervashidze, N., Schweitzer, P., Leeuwen, E. J. v., Mehlhorn, K., and Borgwardt, K. M. (2011). Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(Sep):2539–2561.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46(1-2):159–216.

Socher, R., Huval, B., Manning, C. D., and Ng, A. Y. (2012). Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing (EMNLP) and Computational Natural Language Learning (CNLL)*, pages 1201–1211. Association for Computational Linguistics.

Socher, R., Lin, C. C., Manning, C., and Ng, A. Y. (2011a). Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 129–136.

Socher, R., Pennington, J., Huang, E. H., Ng, A. Y., and Manning, C. D. (2011b). Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 151–161. Association for Computational Linguistics.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1631–1642.

Spelke, E. S., Breinlinger, K., Macomber, J., and Jacobson, K. (1992). Origins of knowledge. *Psychological review*, 99(4):605.

Spelke, E. S. and Kinzler, K. D. (2007). Core knowledge. *Developmental Science*, 10(1):89–96.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Sukhbaatar, S., Fergus, R., et al. (2016). Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, pages 2244–2252.

Sukhbaatar, S., Weston, J., Fergus, R., et al. (2015). End-to-end memory networks. In *Advances in Neural Information Processing Systems*, pages 2440–2448.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 4, page 12.

Tai, K. S., Socher, R., and Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. (2015). Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. International World Wide Web Conferences Steering Committee.

Tenenbaum, J. B., Griffiths, T. L., and Kemp, C. (2006). Theory-based bayesian models of inductive learning and reasoning. *Trends in Cognitive Sciences*, 10(7):309–318.

Tenenbaum, J. B., Kemp, C., Griffiths, T. L., and Goodman, N. D. (2011). How to grow a mind: Statistics, structure, and abstraction. *Science*, 331(6022):1279–1285.

Toyer, S., Trevizan, F., Thiebaux, S., and Xie, L. (2017). Action schema networks: Generalised policies with deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Ullman, T. D., Spelke, E., Battaglia, P., and Tenenbaum, J. B. (2017). Mind games: Game engines as an architecture for intuitive physics. *Trends in Cognitive Sciences*, 21(9):649–665.

van Steenkiste, S., Chang, M., Greff, K., and Schmidhuber, J. (2018). Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph attention networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Wang, J. X., Kurth-Nelson, Z., Kumaran, D., Tirumala, D., Soyer, H., Leibo, J. Z., Hassabis, D., and Botvinick, M. (2018a). Prefrontal cortex as a meta-reinforcement learning system. *Nature neuroscience*, page 1.

Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., and Botvinick, M. (2016). Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*.

Wang, T., Liao, R., Ba, J., and Fidler, S. (2018b). Nervenet: Learning structured policy with graph neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Wang, X., Girshick, R., Gupta, A., and He, K. (2018c). Non-local neural networks. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. (2018d). Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829*.

Watters, N., Zoran, D., Weber, T., Battaglia, P., Pascanu, R., and Tacchetti, A. (2017). Visual interaction networks: Learning a physics simulator from video. In *Advances in Neural Information Processing Systems*, pages 4542–4550.

Wu, J., Lu, E., Kohli, P., Freeman, B., and Tenenbaum, J. (2017). Learning to see physics via visual de-animation. In *Advances in Neural Information Processing Systems*, pages 152–163.

Yoon, K., Liao, R., Xiong, Y., Zhang, L., Fetaya, E., Urtasun, R., Zemel, R., and Pitkow, X. (2018). Inference in probabilistic graphical models by graph neural networks. In *Workshops at the International Conference on Learning Representations (ICLR)*.

You, J., Ying, R., Ren, X., Hamilton, W. L., and Leskovec, J. (2018). GraphRNN: A deep generative model for graphs. *arXiv preprint arXiv:1802.08773*.

Yuille, A. L. and Liu, C. (2018). Deep nets: What have they ever done for vision? *arXiv preprint arXiv:1805.04025*.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. (2017). Deep sets. In *Advances in Neural Information Processing Systems*, pages 3394–3404.

Zambaldi, V., Raposo, D., Santoro, A., Bapst, V., Li, Y., Babuschkin, I., Tuyls, K., Reichert, D., Lillicrap, T., Lockhart, E., Shanahan, M., Langston, V., Pascanu, R., Botvinick, M., Vinyals, O., and Battaglia, P. (2018). Relational deep reinforcement learning. *arXiv preprint arXiv*.

Zhang, A., Lerer, A., Sukhbaatar, S., Fergus, R., and Szlam, A. (2018). Composable planning with attributes. *arXiv preprint arXiv:1803.00512*.

Zügner, D., Akbarnejad, A., and Günnemann, S. (2018). Adversarial Attacks on Neural Networks for Graph Data. *arXiv preprint arXiv:1805.07984*.

Appendix: Formulations of additional models

In this appendix we give more examples of how published networks can fit in the frame defined by Equation 1.

Interaction networks

Interaction Networks (Battaglia et al., 2016; Watters et al., 2017) and the Neural Physics Engine Chang et al. (2017) use a full GN but for the absence of the global to update the edge properties:

$$\begin{aligned}
\phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right) &\coloneqq f^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}\right) &&= \mathrm{NN}_e\left(\left[\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}\right]\right) \\
\phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right) &\coloneqq f^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right) &&= \mathrm{NN}_v\left(\left[\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right]\right) \\
\rho^{e\to v}\left(E'_i\right) &\coloneqq &&= \sum_{\{k:\, r_k=i\}} \mathbf{e}'_k
\end{aligned}$$

That work also included an extension to the above formulation which output global, rather than per-node, predictions:

$$\begin{aligned}
\phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right) &\coloneqq f^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}\right) &&= \mathrm{NN}_e\left(\left[\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}\right]\right) \\
\phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right) &\coloneqq f^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right) &&= \mathrm{NN}_v\left(\left[\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right]\right) \\
\phi^u\left(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}\right) &\coloneqq f^u\left(\bar{\mathbf{v}}', \mathbf{u}\right) &&= \mathrm{NN}_u\left(\left[\bar{\mathbf{v}}', \mathbf{u}\right]\right) \\
\rho^{v\to g}\left(V'\right) &\coloneqq &&= \sum_i \mathbf{v}'_i
\end{aligned}$$

Non-pairwise interactions

Gated Graph Sequence Neural Networks (GGS-NN) (Li et al., 2016) use a slightly generalized formulation where each edge has an attached type $t_k \in \{1, .., T\}$, and the updates are:

$$\begin{aligned}
\phi^e\left(\left(\mathbf{e}_k, t_k\right), \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right) &\coloneqq f^e\left(\mathbf{e}_k, \mathbf{v}_{s_k}\right) &&= \mathrm{NN}_{e,t_k}\left(\mathbf{v}_{s_k}\right) \\
\phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right) &\coloneqq f^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i\right) &&= \mathrm{NN}_v\left(\left[\bar{\mathbf{e}}'_i, \mathbf{v}_i\right]\right) \\
\rho^{e\to v}\left(E'_i\right) &\coloneqq &&= \sum_{\{k:\, r_k=i\}} \mathbf{e}'_k
\end{aligned}$$

These updates are applied recurrently (the NN_v is a GRU (Cho et al., 2014)), followed by a global decoder which computes a weighted sum of embedded final node states. Here each NN_{e,t_k} is a neural network with specific parameters.

CommNet (Sukhbaatar et al., 2016) (in the slightly more general form described by (Hoshen, 2017)) uses:

$$\begin{aligned}
\phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right) &\coloneqq f^e\left(\mathbf{v}_{s_k}\right) &&= \mathrm{NN}_e\left(\mathbf{v}_{s_k}\right) \\
\phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right) &\coloneqq f^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i\right) &&= \mathrm{NN}_v\left(\left[\bar{\mathbf{e}}'_i, \mathrm{NN}_{v'}\left(\mathbf{v}_i\right)\right]\right) \\
\rho^{e\to v}\left(E'_i\right) &\coloneqq &&= \frac{1}{|E'_i|} \sum_{\{k:\, r_k=i\}} \mathbf{e}'_k
\end{aligned}$$

Attention-based approaches

The various attention-based approaches use a ϕ^e which is factored into a scalar pairwise-interaction function which returns the unnormalized attention term, denoted $\alpha^e\left(\mathbf{v}_{r_k}, \mathbf{v}_{s_k}\right) = a'_k$, and a vector-valued non-pairwise term, denoted $\beta^e\left(\mathbf{v}_{s_k}\right) = \mathbf{b}'_k$,

$$\phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right) \coloneqq f^e\left(\mathbf{v}_{r_k}, \mathbf{v}_{s_k}\right) = \left(\alpha^e\left(\mathbf{v}_{r_k}, \mathbf{v}_{s_k}\right),\ \beta^e\left(\mathbf{v}_{s_k}\right)\right) = \left(a'_k, \mathbf{b}'_k\right) = \mathbf{e}'_k$$

The single-headed self-attention (SA) in the Transformer architecture (Vaswani et al., 2017), implements the non-local formulation as:

$$\begin{aligned}
\alpha^e\left(\mathbf{v}_{r_k}, \mathbf{v}_{s_k}\right) &= \exp\left(\mathrm{NN}_{\alpha^{\text{query}}}\left(\mathbf{v}_{r_k}\right)^\intercal \cdot \mathrm{NN}_{\alpha^{\text{key}}}\left(\mathbf{v}_{s_k}\right)\right) \\
\beta^e\left(\mathbf{v}_{s_k}\right) &= \mathrm{NN}_\beta\left(\mathbf{v}_{s_k}\right) \\
\phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right) \coloneqq f^v\left(\bar{\mathbf{e}}'_i\right) &= \mathrm{NN}_v\left(\bar{\mathbf{e}}'_i\right)
\end{aligned}$$

where $\mathrm{NN}_{\alpha^{\text{query}}}$, $\mathrm{NN}_{\alpha^{\text{key}}}$, and NN_β are again neural network functions with different parameters and possibly different architectures. They also use a multi-headed version which computes N_h parallel $\bar{\mathbf{e}}'^h_i$ using different $\mathrm{NN}_{\alpha^{\text{query}}_h}$, $\mathrm{NN}_{\alpha^{\text{key}}_h}$, NN_{β_h}, where h indexes the different parameters. These are passed to f^v and concatenated:

$$f^v\left(\left\{\bar{\mathbf{e}}'^h_i\right\}_{h=1\ldots N^h}\right) = \mathrm{NN}_v\left(\left[\bar{\mathbf{e}}'^1_i, \ldots, \bar{\mathbf{e}}'^{N_h}_i\right]\right)$$

Vertex Attention Interaction Networks (Hoshen, 2017) are very similar to single-headed SA, but use Euclidean distance for the attentional similarity metric, with shared parameters across the attention inputs' embeddings, and also use the input node feature in the node update function,

$$\begin{aligned}
\alpha^e\left(\mathbf{v}_{r_k}, \mathbf{v}_{s_k}\right) &= \exp\left(-\left\|\mathrm{NN}_\alpha\left(\mathbf{v}_{r_k}\right) - \mathrm{NN}_\alpha\left(\mathbf{v}_{s_k}\right)\right\|^2\right) \\
\beta^e\left(\mathbf{v}_{s_k}\right) &= \mathrm{NN}_\beta\left(\mathbf{v}_{s_k}\right) \\
\phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right) \coloneqq f^v\left(\bar{\mathbf{e}}'_i\right) &= \mathrm{NN}_v\left(\left[\bar{\mathbf{e}}'_i, \mathbf{v}_i\right]\right)
\end{aligned}$$

Graph Attention Networks (Veličković et al., 2018) are also similar to multi-headed SA, but use a neural network as the attentional similarity metric, with shared parameters across the attention inputs' embeddings:

$$\begin{aligned}
\alpha^e\left(\mathbf{v}_{r_k}, \mathbf{v}_{s_k}\right) &= \exp\left(\mathrm{NN}_{\alpha'}\left(\left[\mathrm{NN}_\alpha\left(\mathbf{v}_{r_k}\right), \mathrm{NN}_\alpha\left(\mathbf{v}_{s_k}\right)\right]\right)\right) \\
\beta^e\left(\mathbf{v}_{s_k}\right) &= \mathrm{NN}_\beta\left(\mathbf{v}_{s_k}\right) \\
\phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right) \coloneqq f^v\left(\left\{\bar{\mathbf{e}}'^h_i\right\}_{h=1\ldots N_h}\right) &= \mathrm{NN}_v\left(\left[\bar{\mathbf{e}}'^1_i, \ldots, \bar{\mathbf{e}}'^{N_h}_i\right]\right)
\end{aligned}$$

Stretching beyond the specific non-local formulation, Shaw et al. (2018) extended multi-headed SA with relative position encodings. "Relative" refers to an encoding of the spatial distance between nodes in a sequence or other signal in a metric space. This can be expressed in GN language as an edge attribute $\mathbf{e}_k$, and replacing the $\beta^e\left(\mathbf{v}_{s_k}\right)$ from multi-headed SA above with:

$$\beta^e\left(\mathbf{e}_k, \mathbf{v}_{s_k}\right) = \mathrm{NN}_e\left(\mathbf{v}_{s_k}\right) + \mathbf{e}_k$$

Belief Propagation embeddings

Finally, we briefly summarize how the general "structure2vec" algorithm of Dai et al. (2016) can fit into our framework. In order to do so, we need to slightly modify our main Equation 1, i.e.:

$$
\begin{aligned}
\bar{\boldsymbol{\epsilon}}_k = \rho\left(\{\mathbf{e}_l\}_{\substack{s_l=r_k \\ r_l \neq s_k}}\right) &\coloneqq \sum_{\substack{r_l=s_k \\ s_l \neq r_k}} \mathbf{e}_l \\
\mathbf{e}'_k = \phi^e\left(\bar{\boldsymbol{\epsilon}}_k\right) &\coloneqq f(\bar{\boldsymbol{\epsilon}}_k) = \mathrm{NN}(\bar{\boldsymbol{\epsilon}}_k) \\
\bar{\mathbf{e}}'_i = \rho\left(\{\mathbf{e}'_k\}_{r_k=i}\right) &\coloneqq \sum_{\{k:\, r_k=i\}} \mathbf{e}_k \\
\mathbf{v}'_i = \phi^v\left(\bar{\mathbf{e}}'_i\right) &\coloneqq f(\bar{\mathbf{e}}'_i) = \mathrm{NN}(\bar{\mathbf{e}}'_i)
\end{aligned}
$$

Edges' features now takes the meaning of "message" between their receiver and sender; note that there is only one set of parameters to learn for both the edges and nodes updates.