Relational Inductive Biases, Deep Learning, and Graph Networks

Hyungu Kahng

2019.01.11

@DMQA Open Seminar

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. As a companion to this paper, we have also released an open-source software library for building graph networks, with demonstrations of how to use them in practice.

Contents

Inductive Bias

Definitions

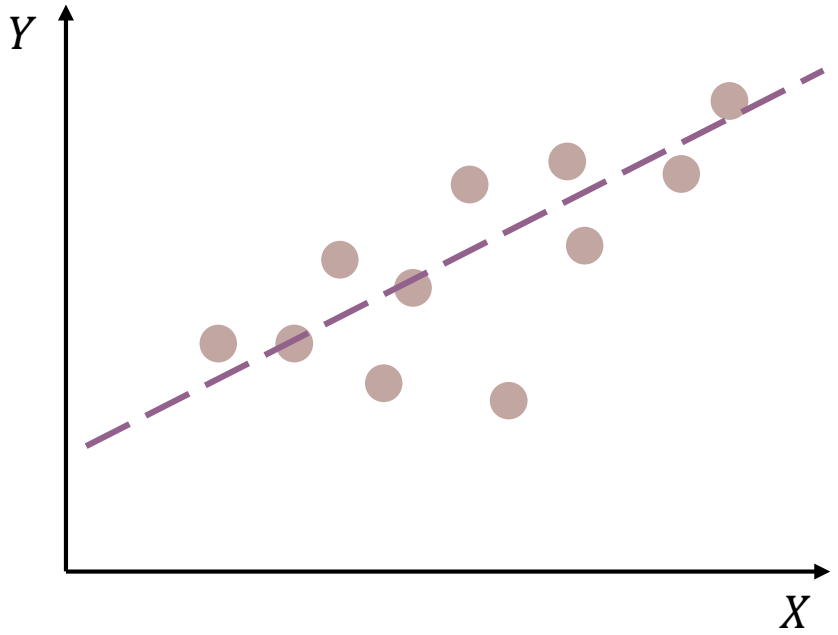
An inductive bias allows a learning algorithm to prioritize one solution over another, independent of the observed data.

The inductive bias of a learning algorithm is the set of assumptions that the learner uses to predict outputs given inputs that it has not encountered.

Inductive Bias

in linear regression

- The relationship between predictor variables X and the target Y can be expressed in linear combinations, and errors should be minimal under a quadratic penalty.



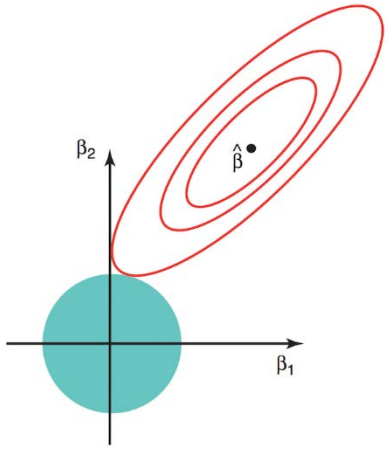
$$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_i + \hat{\epsilon}_i$$

$$\min_{\beta} \sum_{i=1}^{N} (Y_i - \hat{Y}_i)^2$$

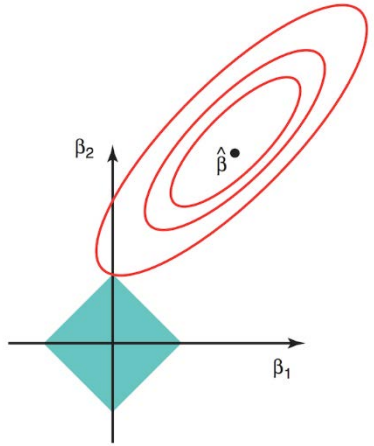
Inductive Bias

in ridge & lasso regression

- Prefers solutions with smaller regression coefficients



$$\min_{\beta} \sum_{i=1}^{N} (Y_i - \hat{Y}_i)^2 + \lambda \sum_{p=1}^{P} |\beta_p|^2$$

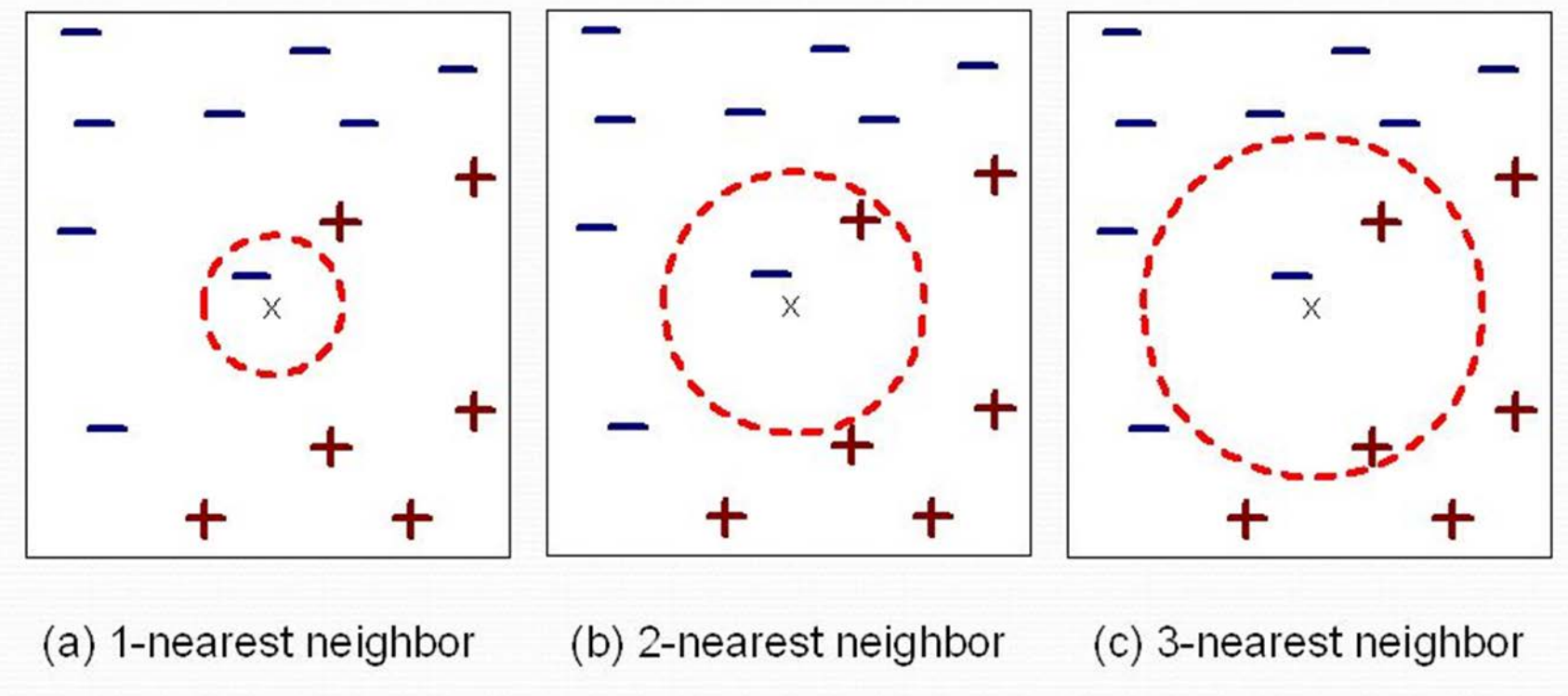


$$\min_{\beta} \sum_{i=1}^{N} (Y_i - \hat{Y}_i)^2 + \lambda \sum_{p=1}^{P} |\beta_p|$$

Inductive Bias

in k-nearest neighbors

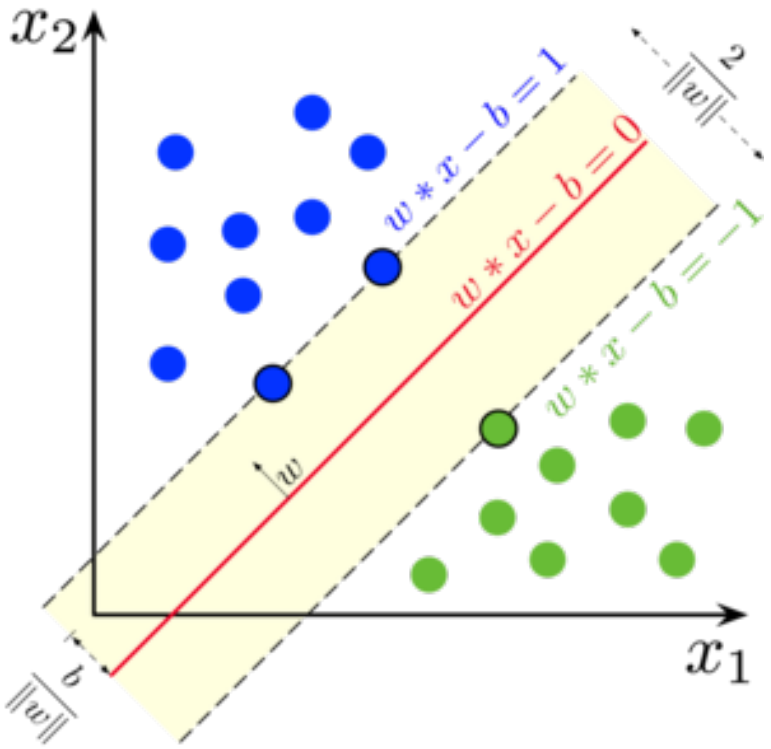
- The classification of an observation will be most similar to the classification of other instances that are nearby



Inductive Bias

in support vector machines

- When drawing a boundary between two classes, attempt to maximize the width of the boundary.

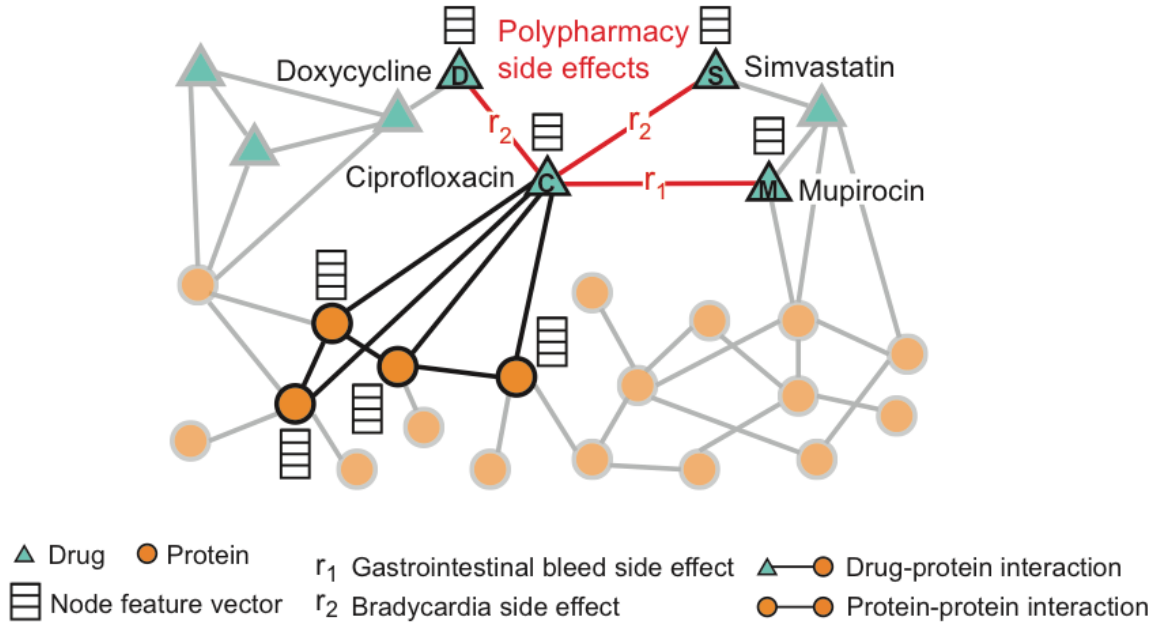


$$\frac{1}{2}\|w\|^2 + C\sum_i \max(0, 1 - y_i(w^\intercal x_i + b))$$

Relational Inductive Bias

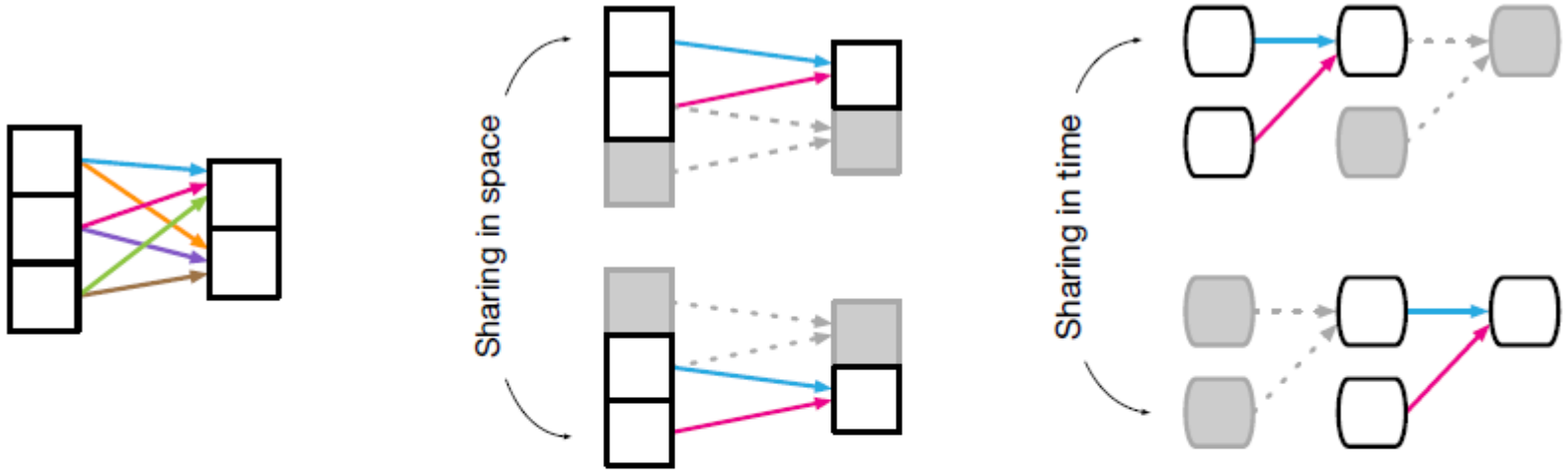
Definition

- Refers to inductive biases which impose constraints on relationships and interactions among entities in a learning process.



Relational Inductive Bias

in neural networks



(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.

Relational Inductive Bias

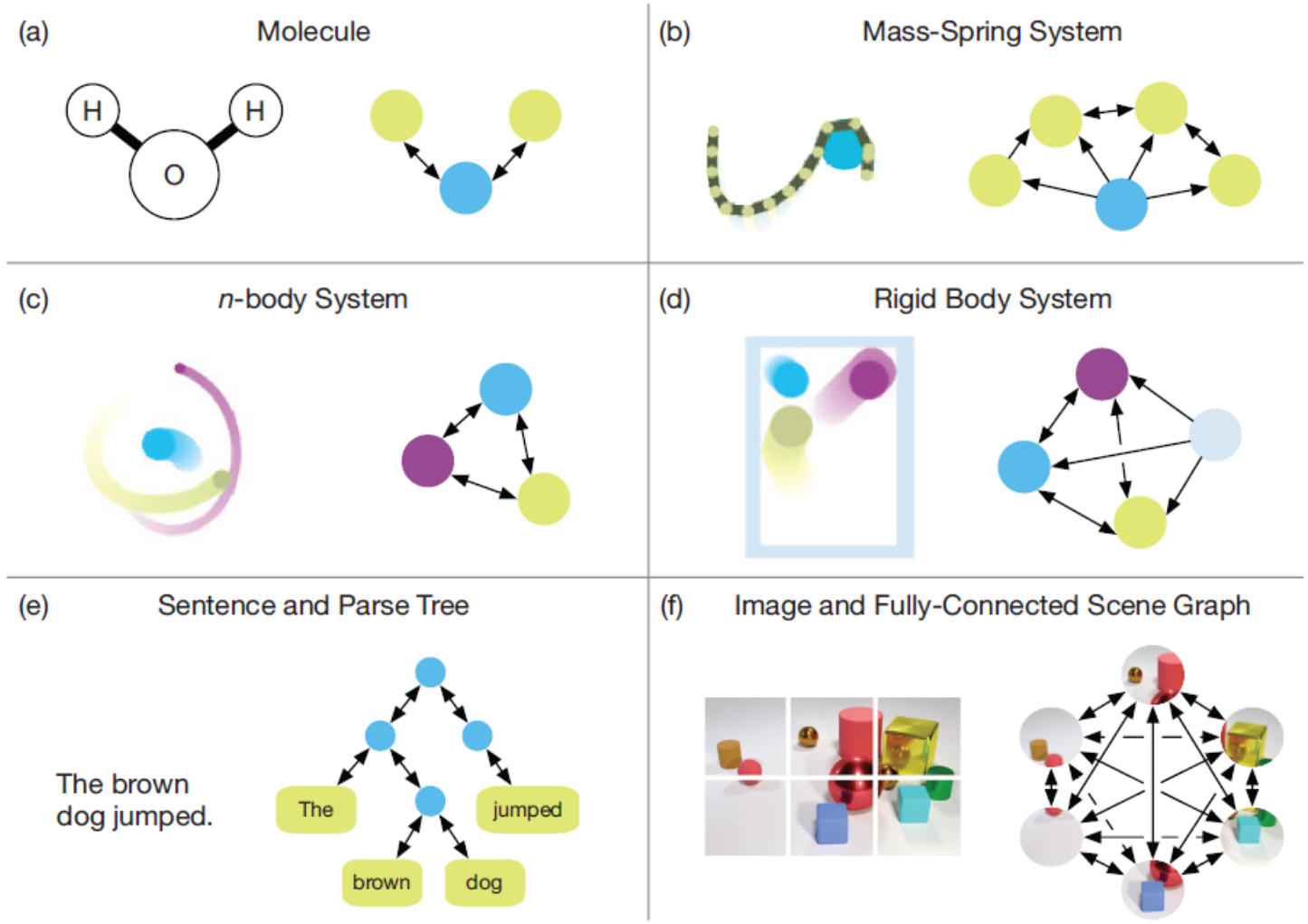
in neural networks

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.

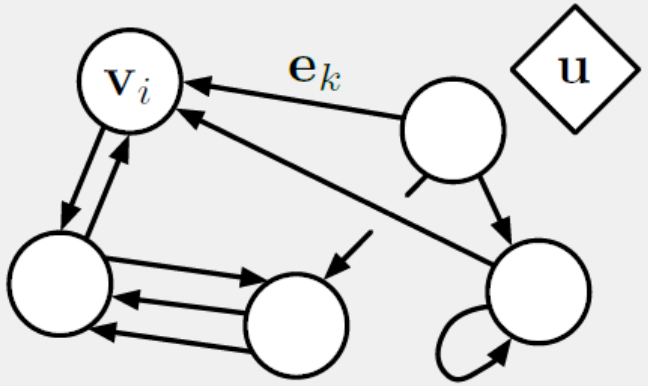
Graph Neural Networks

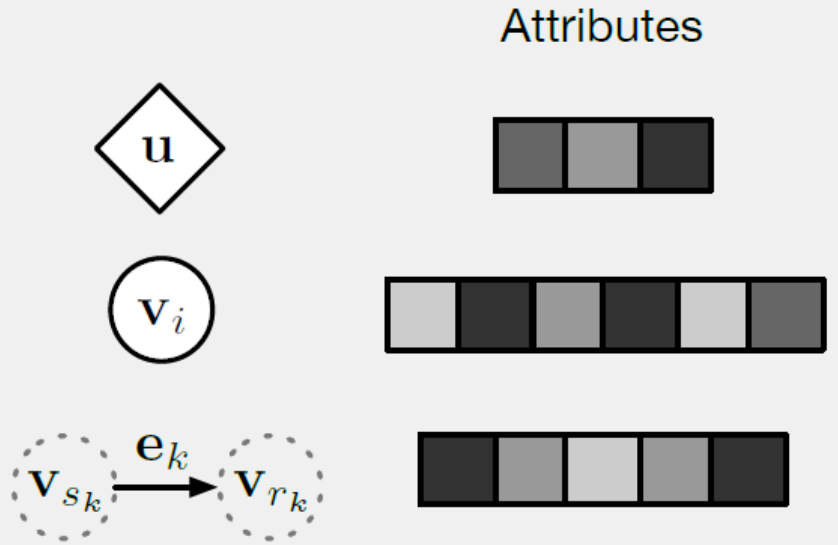
Different graph representations



Graph Neural Networks

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:

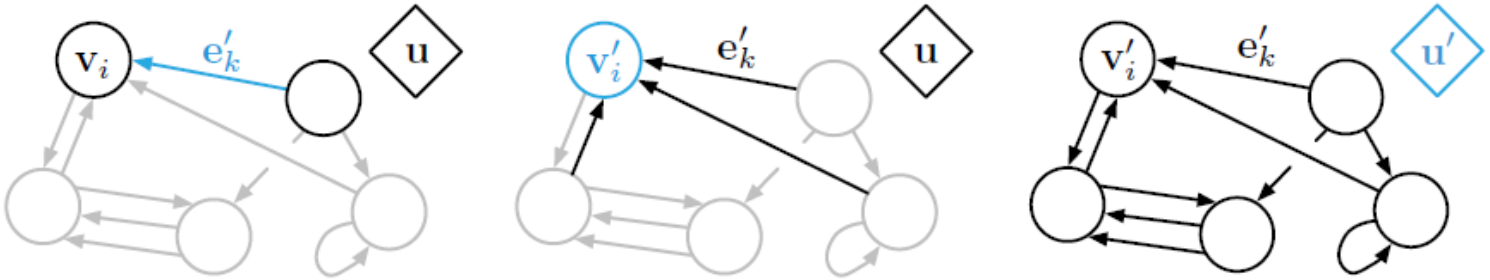
Graph Neural Networks

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



(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.

Graph Neural Networks

Pseudo code

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 $E'_i = \{(\mathbf{e}'_k, r_k, s_k)\}_{r_k=i,\ k=1:N^e}$

    $\bar{\mathbf{e}}'_i \leftarrow \rho^{e\rightarrow v}\left(E'_i\right)$ ▷ 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 $V' = \{\mathbf{v}'\}_{i=1:N^v}$

  let $E' = \{(\mathbf{e}'_k, r_k, s_k)\}_{k=1:N^e}$

  $\bar{\mathbf{e}}' \leftarrow \rho^{e\rightarrow u}\left(E'\right)$ ▷ 4. Aggregate edge attributes globally

  $\bar{\mathbf{v}}' \leftarrow \rho^{v\rightarrow u}\left(V'\right)$ ▷ 5. Aggregate node attributes globally

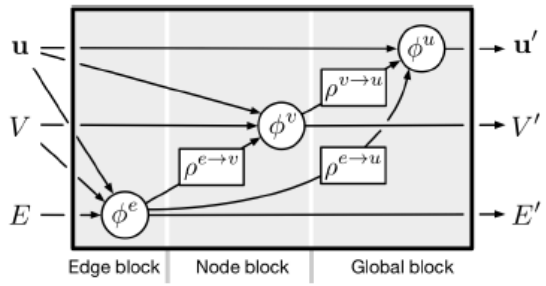
  $\mathbf{u}' \leftarrow \phi^u\left(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}\right)$ ▷ 6. Compute updated global attribute

  return $(E', V', \mathbf{u}')$

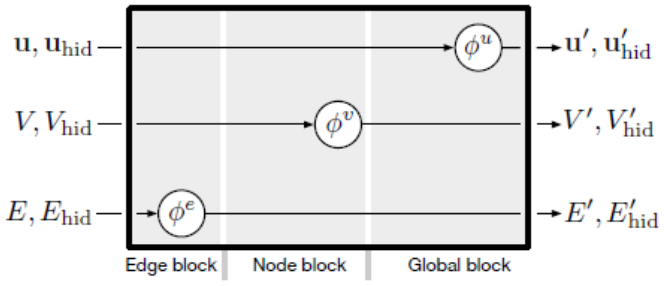
end function

Graph Neural Networks

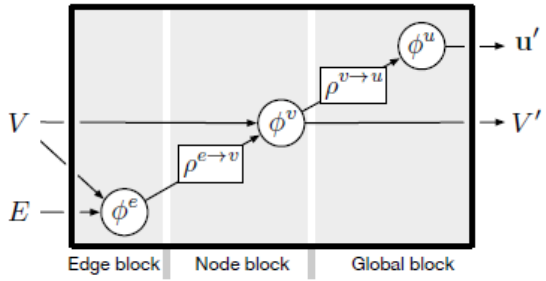
Different GN block configurations



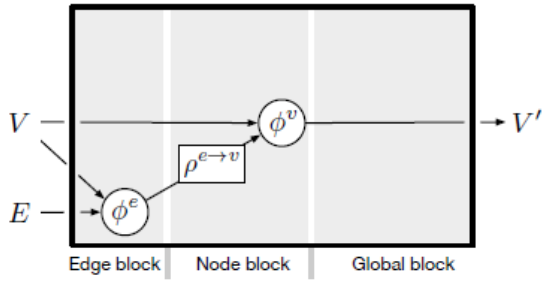
(a) Full GN block



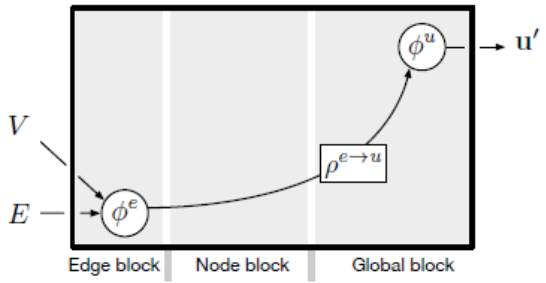
(b) Independent recurrent block



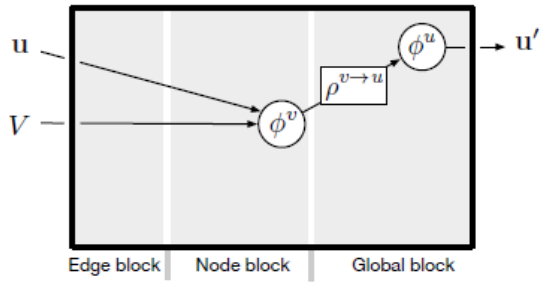
(c) Message-passing neural network



(d) Non-local neural network



(e) Relation network



(f) Deep set

Graph Neural Networks

Message passing

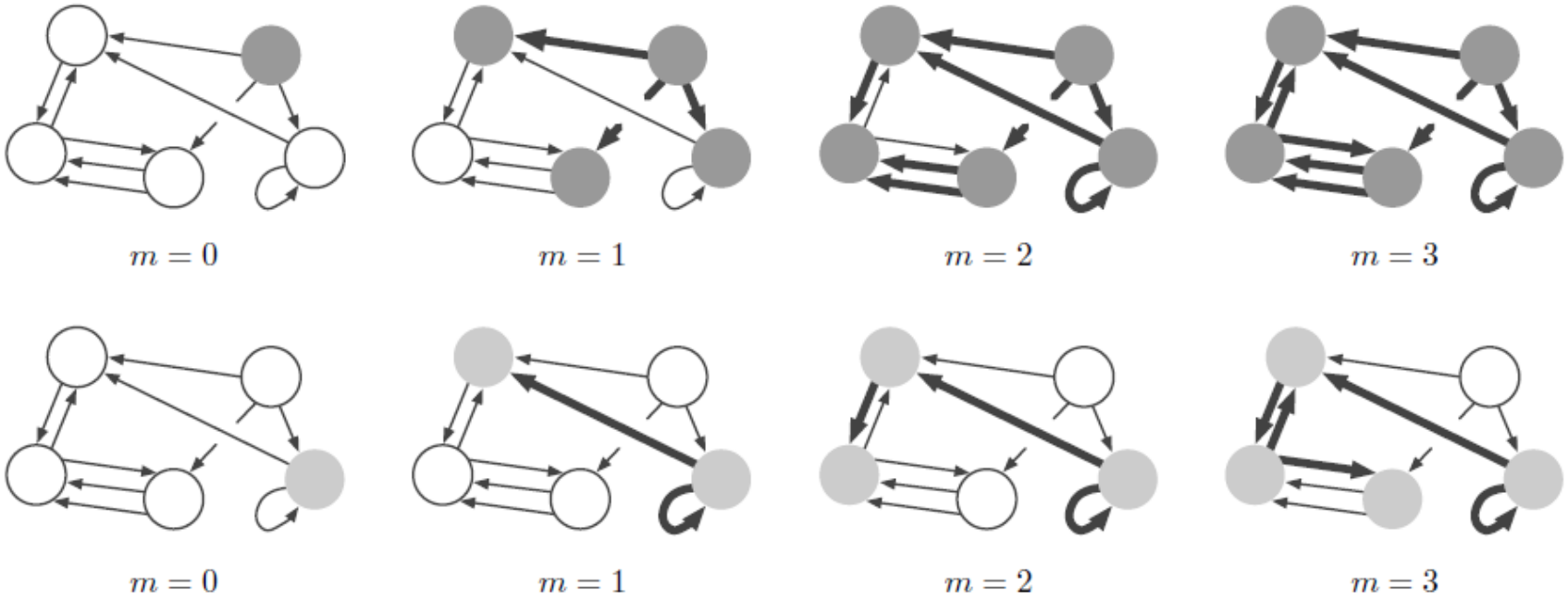
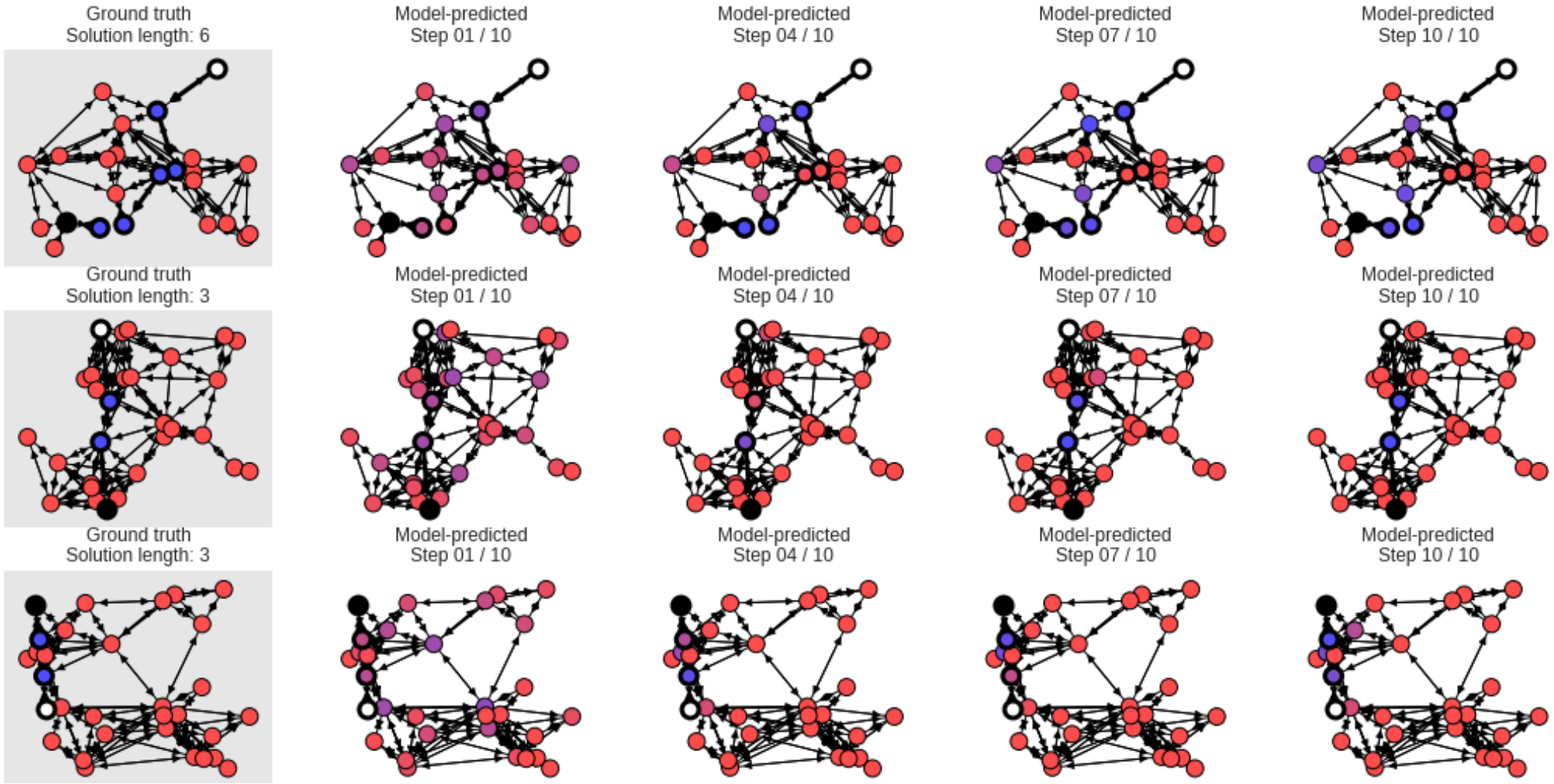


Figure 7: Example of message passing. Each row highlights the information that diffuses through the graph starting from a particular node. In the top row, the node of interest is in the upper right; in the bottom row, the node of interest is in the bottom right. Shaded nodes indicate how far information 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).

GNN Applications

Shortest path prediction



GNN Applications

van den Berg, R. et al., Graph Convolutional Matrix Completion, 2017

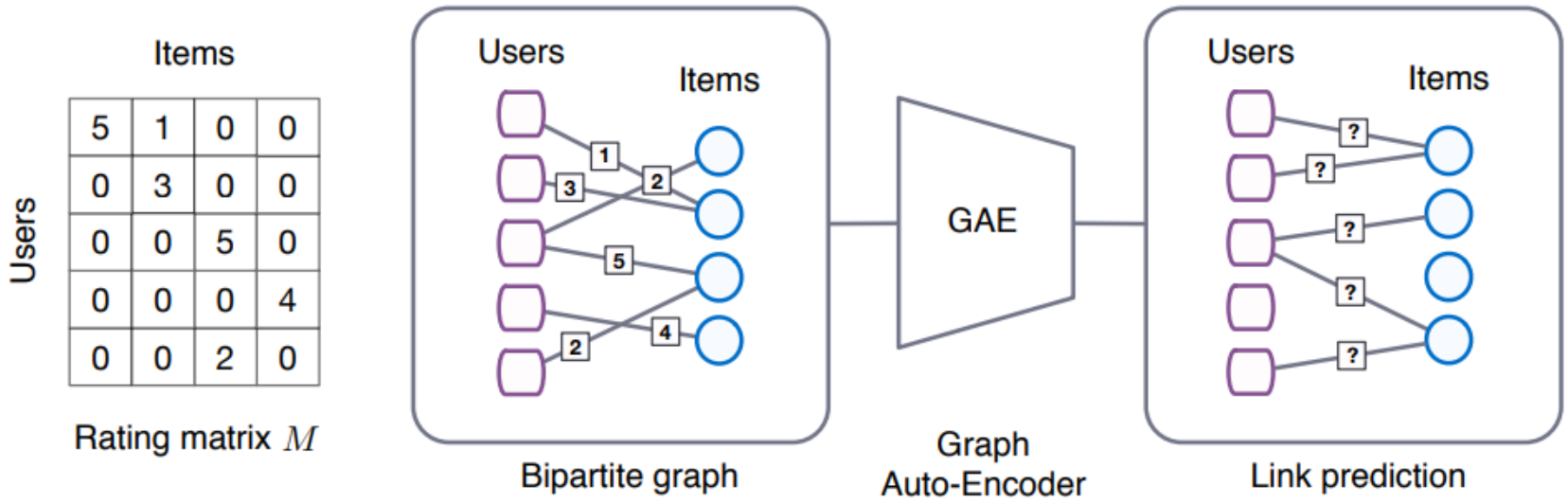


Figure 1: *Left*: Rating matrix M with entries that correspond to user-item interactions (ratings between 1-5) or missing observations (0). *Right*: User-item interaction graph with bipartite structure. Edges correspond to interaction events, numbers on edges denote the rating a user has given to a particular item. The matrix completion task (i.e. predictions for unobserved interactions) can be cast as a link prediction problem and modeled using an end-to-end trainable graph auto-encoder.

van den Berg, R., Kipf, T. N., & Welling, M. (2017). Graph Convolutional Matrix Completion. stat, 1050, 7.

GNN Applications

Gilmer J. et al., Neural Message Passing for Quantum Chemistry, 2017

- Predicting various chemical properties of organic molecules (13 regression tasks)

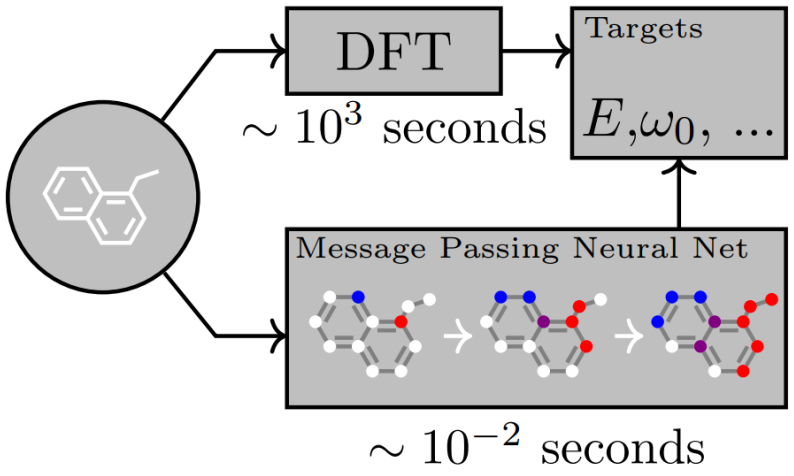


Figure 1. A Message Passing Neural Network predicts quantum properties of an organic molecule by modeling a computationally expensive DFT calculation.

Table 2. Comparison of Previous Approaches (left) with MPNN baselines (middle) and our methods (right)

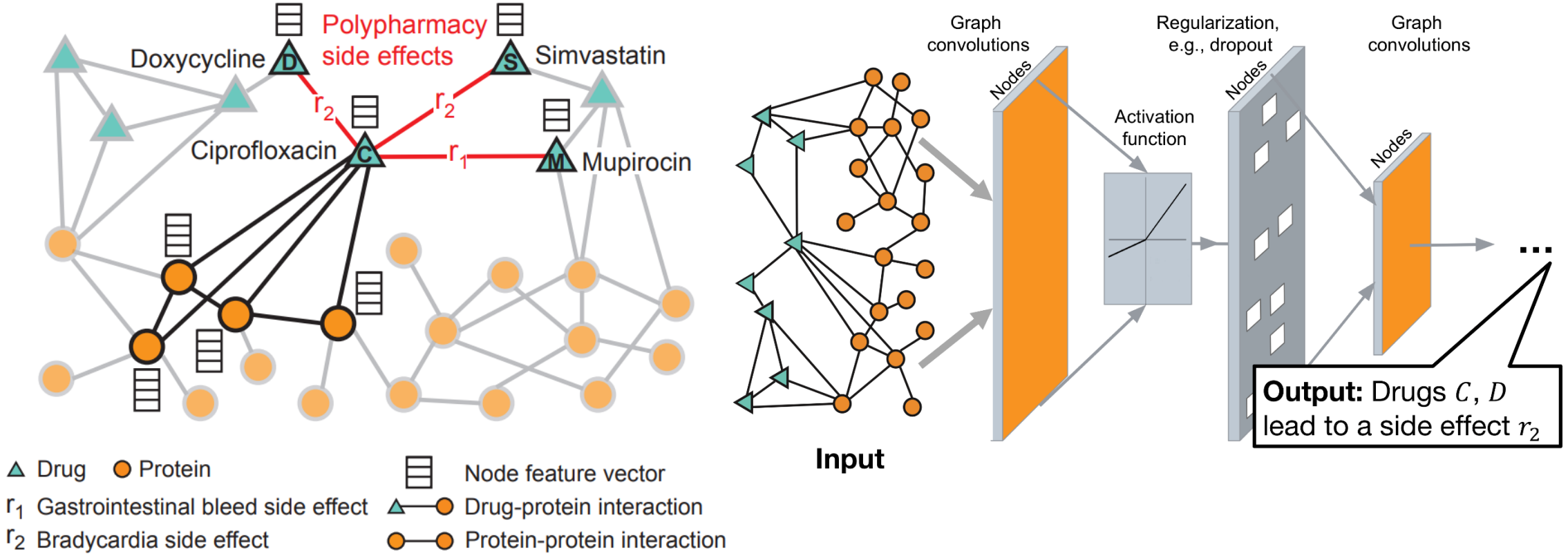
Target	BAML	BOB	CM	ECFP4	HDAD	GC	GG-NN	DTNN	enn-s2s	enn-s2s-ens5
mu	4.34	4.23	4.49	4.82	3.34	0.70	1.22	-	**0.30**	0.20
alpha	3.01	2.98	4.33	34.54	1.75	2.27	1.55	-	**0.92**	0.68
HOMO	2.20	2.20	3.09	2.89	1.54	1.18	1.17	-	**0.99**	0.74
LUMO	2.76	2.74	4.26	3.10	1.96	1.10	1.08	-	**0.87**	0.65
gap	3.28	3.41	5.32	3.86	2.49	1.78	1.70	-	**1.60**	1.23
R2	3.25	0.80	2.83	90.68	1.35	4.73	3.99	-	**0.15**	0.14
ZPVE	3.31	3.40	4.80	241.58	1.91	9.75	2.52	-	**1.27**	1.10
U0	1.21	1.43	2.98	85.01	0.58	3.02	0.83	-	**0.45**	0.33
U	1.22	1.44	2.99	85.59	0.59	3.16	0.86	-	**0.45**	0.34
H	1.22	1.44	2.99	86.21	0.59	3.19	0.81	-	**0.39**	0.30
G	1.20	1.42	2.97	78.36	0.59	2.95	0.78	.84[2]	**0.44**	0.34
Cv	1.64	1.83	2.36	30.29	0.88	1.45	1.19	-	**0.80**	0.62
Omega	0.27	0.35	1.32	1.47	0.34	0.32	0.53	-	**0.19**	0.15
Average	2.17	2.08	3.37	53.97	1.35	2.59	1.36	-	**0.68**	0.52

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

GNN Applications

Zitnik, M. et al., Modeling Polypharmacy Side Effects with Graph Convolutional Networks, 2018

- Multirelational link prediction to model polypharmacy side effects
- 964 different types of edges (one for each side effect type)



Zitnik, M., Agrawal, M., & Leskovec, J. (2018). Modeling polypharmacy side effects with graph convolutional networks. arXiv preprint arXiv:1802.00543.

GNN Applications

Wang, X. et al., Non-local Neural Networks, 2018

Figure 3. Examples of the behavior of a non-local block in res_3 computed by a 5-block non-local model trained on Kinetics. These examples are from held-out validation videos. The starting point of arrows represents one $\mathbf{x}_i$, and the ending points represent $\mathbf{x}_j$. The 20 highest weighted arrows for each $\mathbf{x}_i$ are visualized. The 4 frames are from a 32-frame input, shown with a stride of 8 frames. These visualizations show how the model finds related clues to support its prediction.

Wang, X., Girshick, R., Gupta, A., & He, K. (2018, June). Non-local neural networks. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (Vol. 1, No. 3, p. 4).

GNN Applications

Kool, W. et al., Attention, Learn to Solve Routing Problems, 2018 (1/2)

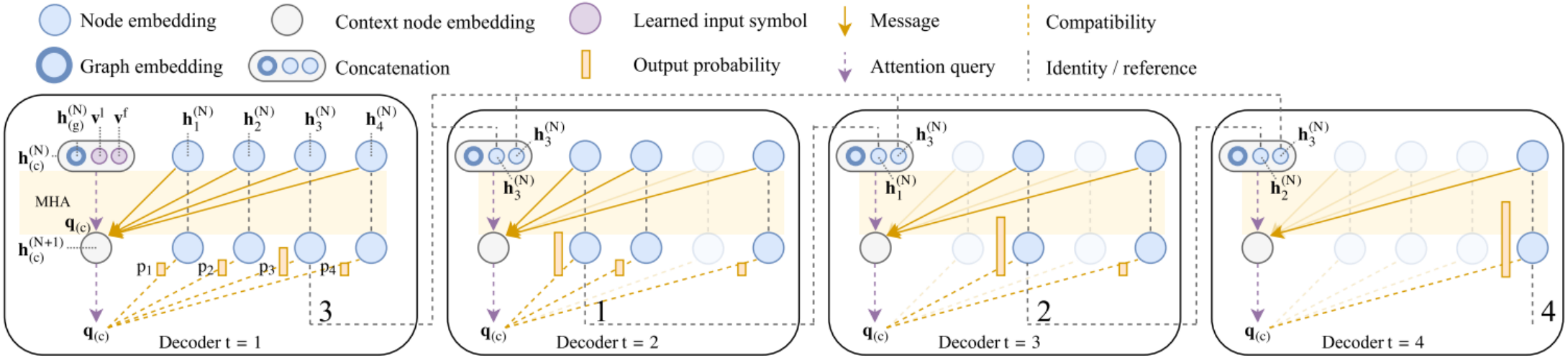


Figure 2: Attention based decoder for the TSP problem. The decoder takes as input the graph embedding and node embeddings. At each time step t, the context consist of the graph embedding and the embeddings of the first and last (previously output) node of the partial tour, where learned placeholders are used if $t = 1$. Nodes that cannot be visited (since they are already visited) are masked. The example shows how a tour $\boldsymbol{\pi} = (3, 1, 2, 4)$ is constructed. Best viewed in color.

Kool, W., van Hoof, H., & Welling, M. (2018). Attention, Learn to Solve Routing Problems!.

GNN Applications

Kool, W. et al., Attention, Learn to Solve Routing Problems, 2018 (2/2)

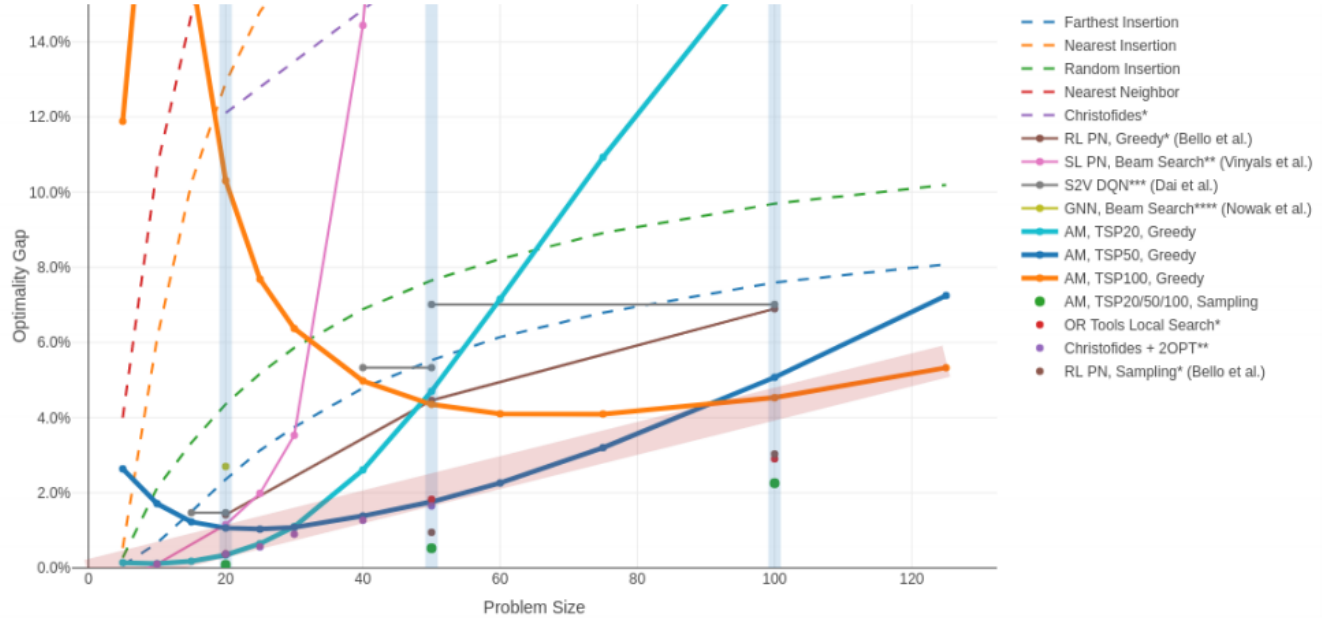


Figure 5: Optimality gap of different methods as a function of problem size $n \in \{5, 10, 15, 20, 25, 30, 40, 50, 60, 75, 100, 125\}$. General baselines are drawn using dashed lines while learned algorithms are drawn with a solid line. Algorithms (general and learned) that perform search or sampling are plotted without connecting lines for clarity. The *, **, *** and **** indicate that values are reported from Bello et al. (2016), Vinyals et al. (2015), Dai et al. (2017) and Nowak et al. (2017) respectively. Best viewed in color.

Kool, W., van Hoof, H., & Welling, M. (2018). Attention, Learn to Solve Routing Problems.

GNN Applications

Zambaldi, V. et al., Deep Reinforcement Learning with Relational Inductive Biases, 2018 (1/4)

Figure 1: Box-World and StarCraft II tasks demand reasoning about entities and their relations.

Zambaldi, V., Raposo, D., Santoro, A., Bapst, V., Li, Y., Babuschkin, I., ... & Shanahan, M. (2018). Deep reinforcement learning with relational inductive biases.

GNN Applications

Zambaldi, V. et al., Deep Reinforcement Learning with Relational Inductive Biases, 2018 (2/4)

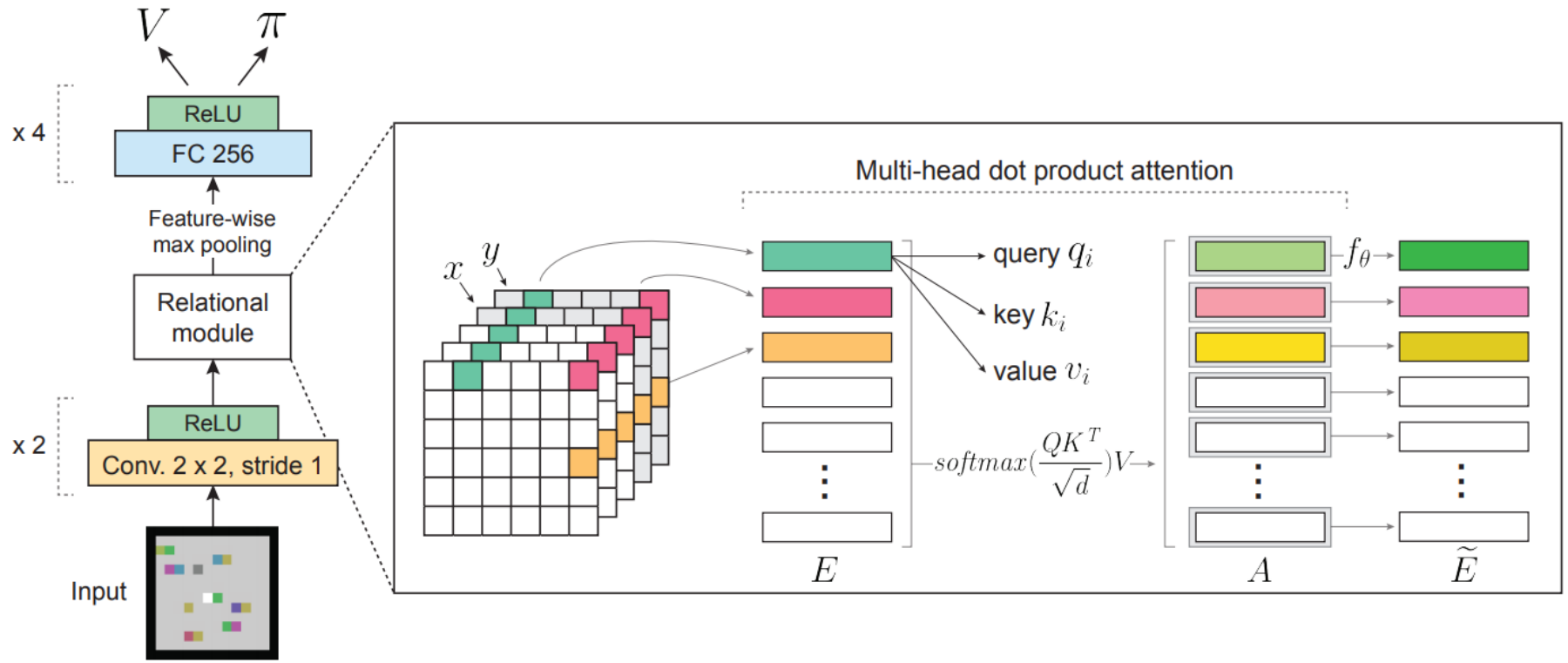


Figure 2: Box-World agent architecture and multi-head dot-product attention. E is a matrix that compiles the entities produced by the visual front-end; f_θ is a multilayer perceptron applied in parallel to each row of the output of an MHDPA step, A, and producing updated entities, $\tilde{E}$.

Zambaldi, V., Raposo, D., Santoro, A., Bapst, V., Li, Y., Babuschkin, I., ... & Shanahan, M. (2018). Deep reinforcement learning with relational inductive biases.

GNN Applications

Zambaldi, V. et al., Deep Reinforcement Learning with Relational Inductive Biases, 2018 (3/4)

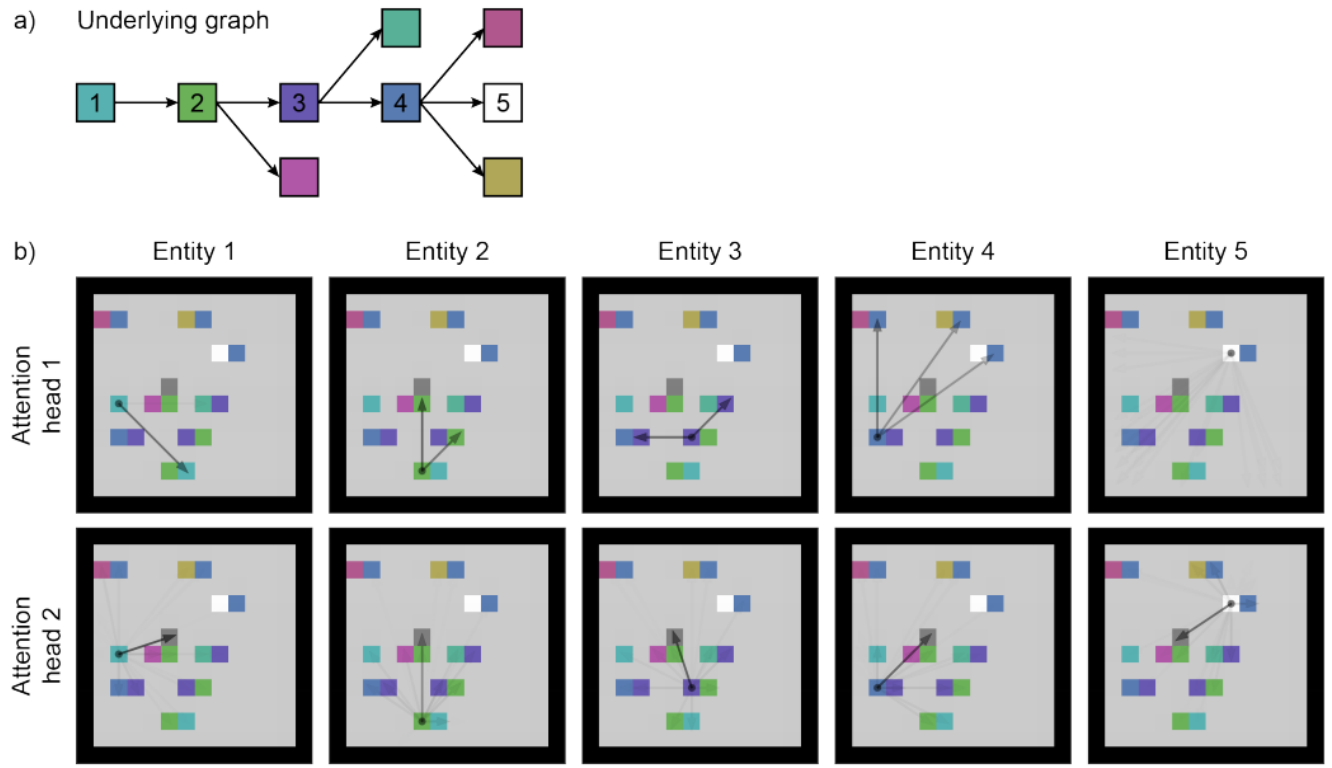


Figure 4: Visualization of attention weights. (a) The underlying graph of one example level; (b) the result of the analysis for that level, using each of the entities along the solution path (1–5) as the *source* of attention. Arrows point to the entities that the *source* is attending to. An arrow's transparency is determined by the corresponding attention weight.

Zambaldi, V., Raposo, D., Santoro, A., Bapst, V., Li, Y., Babuschkin, I., ... & Shanahan, M. (2018). Deep reinforcement learning with relational inductive biases.

GNN Applications

Zambaldi, V. et al., Deep Reinforcement Learning with Relational Inductive Biases, 2018 (4/4)

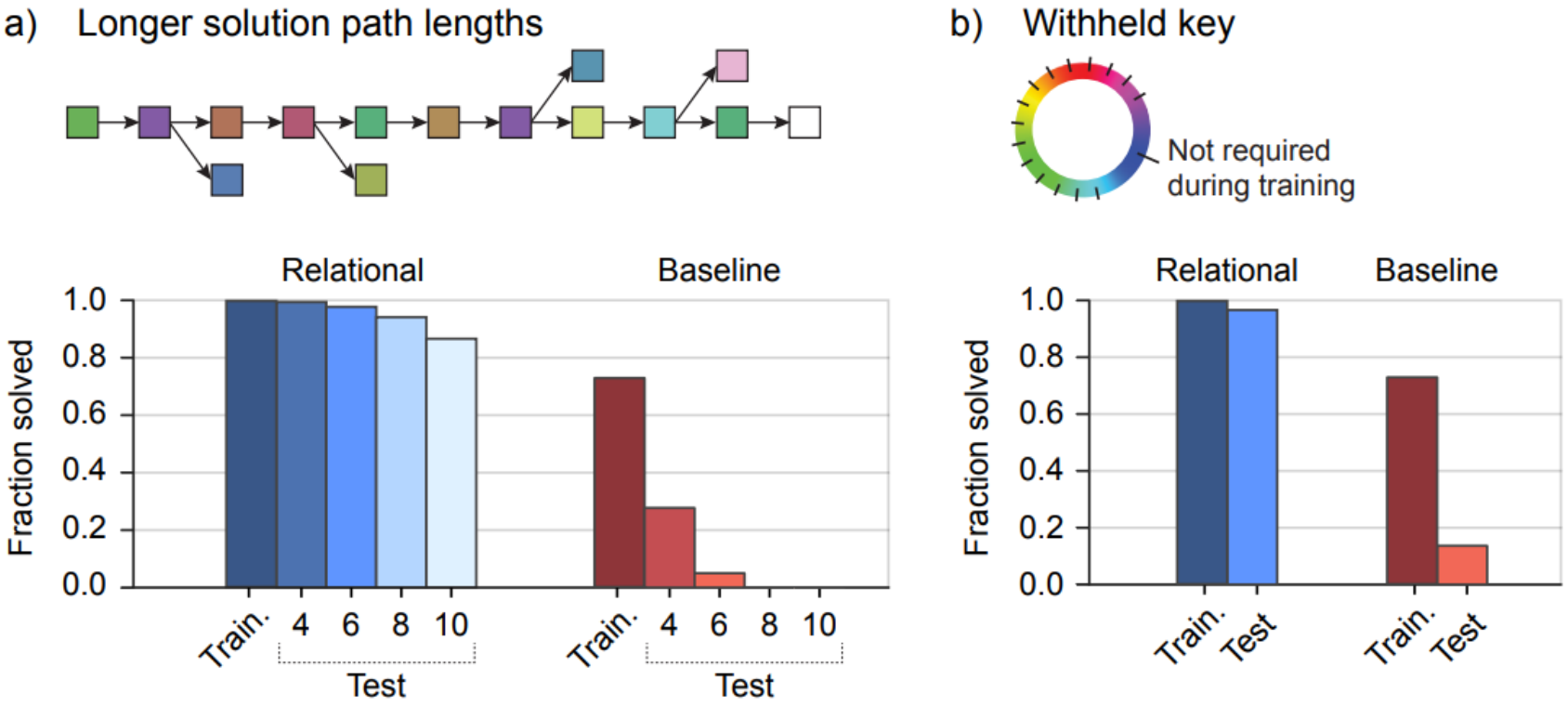


Figure 5: Generalization in Box-World. Zero-shot transfer to levels that required: (a) opening a longer sequence of boxes; (b) using a key-lock combination that was never required during training.

Zambaldi, V., Raposo, D., Santoro, A., Bapst, V., Li, Y., Babuschkin, I., ... & Shanahan, M. (2018). Deep reinforcement learning with relational inductive biases.

Python Implementations

In both Tensorflow and Pytorch

- (Tensorflow, with Sonnet) https://github.com/deepmind/graph_nets
- (Pytorch) https://github.com/rusty1s/pytorch_geometric
- (Pytorch) https://github.com/dmlc/dgl

Discussions

Limitations and future work

- Where do the graphs come from that the computations are held on?
  - Assuming a fully-connected, dense graph is too naïve.
  - Many underlying graph structures are much more sparse.

- Can the graph structures be modified adaptively during the course of computation?
- Can the edges be removed depending on the context of the graph?
- …

Additional Papers

Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon*

Yoshua Bengio[2,3], Andrea Lodi[1,3], and Antoine Prouvost[1,3]

yoshua.bengio@mila.quebec
{andrea.lodi, antoine.prouvost}@polymtl.ca

[1]Canada Excellence Research Chair in Data Science for Decision Making, École Polytechnique de Montréal
[2]Department of Computer Science and Operations Research, Université de Montréal
[3]Mila, Quebec Artificial Intelligence Institute

Abstract

This paper surveys the recent attempts, both from the machine learning and operations research communities, at leveraging machine learning to solve combinatorial optimization problems. Given the hard nature of these problems, state-of-the-art methodologies involve algorithmic decisions that either require too much computing time or are not mathematically well defined. Thus, machine learning looks like a promising candidate to effectively deal with those decisions. We advocate for pushing further the integration of machine learning and combinatorial optimization and detail methodology to do so. A main point of the paper is seeing generic optimization problems as data points and inquiring what is the relevant distribution of problems to use for learning on a given task.

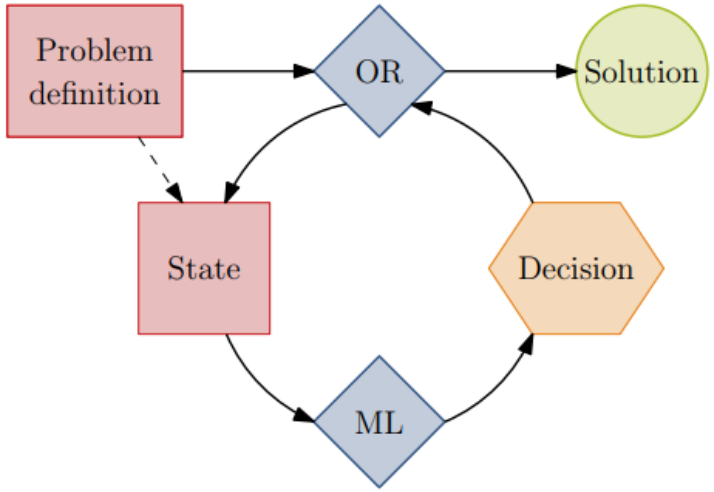


Figure 9: The combinatorial optimization algorithm repeatedly queries the same ML model to make decisions. The ML model takes as input the current state of the algorithm, which may include the problem definition.

Thank you.