Beyond Graph Neural Networks with Lifted Relational Neural Networks

Gustav Sourek, Filip Zelezny, Ondrej Kuzelka

Czech Technical University in Prague

Introduction

We demonstrate a deep learning framework which is inherently based in the highly expressive language of relational logic. The declarative nature of the used language then allows to use compact and elegant learning programs to capture a wide range of advanced neural models, with a particular focus on Graph Neural Networks (GNNs). We show how GNNs can be naturally covered in the framework by specifying the underlying representation propagation rules in the relational logic, and further easily extended towards higher expressiveness in various ways.

Relational Templating of Neural Models

Templating is a form of indirect encoding of neural networks in the language of logic. In its propositional form, this was popularly known as Knowledge-based neural network construction [1], the expressiveness of which was, however, limited to standard neural networks. We updated the concept into relational (FOL) setting with Lifted Relational Neural Networks (LRNNs) [2] to reflect the increased (lifted) language expressiveness in the neural domain, too.

Introduction by Example (GNN)

Consider a simple template for learning with molecules, encoding a generic idea that the representation $(\mathrm{h}(.))$ of a chemical atom (e.g. o_1) is dependent on the atoms adjacent to it. Given that a molecule can be represented by the set of contained atoms (e.g. $\mathrm{a}(\mathrm{o}_1), ..., \mathrm{a}(\mathrm{h}_2)$) and bonds between them (e.g. $\mathrm{b}(\mathrm{o}_1, \mathrm{h}_2)$), we can encode this idea by a following weighted rule:

$$\mathbf{W}_{\mathrm{h}_1} :: \mathrm{h}(\mathrm{X}) \text{ :- } \mathbf{W}_{\mathrm{a}} : \mathrm{a}(\mathrm{Y}),\ \mathbf{W}_{\mathrm{b}} : \mathrm{b}(\mathrm{X},\mathrm{Y}).$$

where X, Y are free variables. Moreover, one might be interested in using the representation of all atoms $(\mathrm{h}(\mathrm{X}))$ for deducing the representation of the whole molecule (q), i.e.:

$$\mathbf{W}_{\mathrm{q}} :: \mathrm{q} \text{ :- } \mathbf{W}_{\mathrm{h}_2} : \mathrm{h}(\mathrm{X}).$$

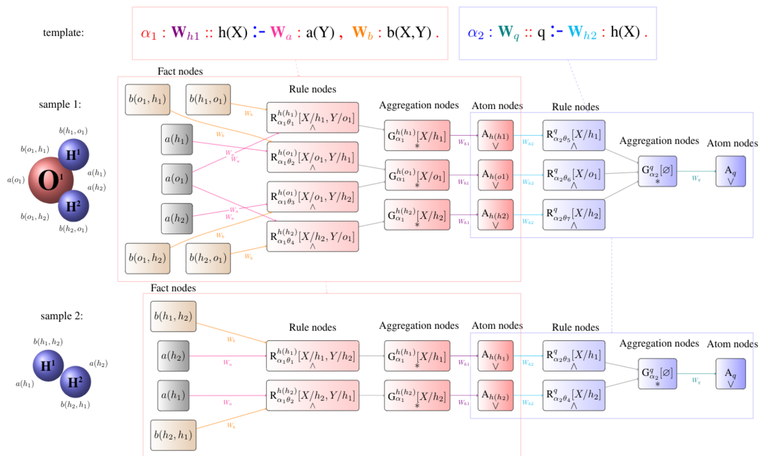


Figure 1: A simple GNN-like template with a "propagation" and a "readout" rule. Upon receiving 2 molecules, 2 neural computation graphs will get (automatically) created by the framework.

To create a neural computation graph $\mathcal{G}_{\mathrm{l}}$ from such a learning template $\mathcal{T}$ and example E_{l}, the engine constructs the *least Herbrand model* $\mathcal{H}$ of $\mathcal{T} \cup \mathrm{E}_{\mathrm{l}}$, which can be done using standard techniques. The derived logical constructs from $\mathcal{H}$ are then projected onto specific node types in the (differentiable) computation graph $\mathcal{G}_{\mathrm{l}}$, as outlined in Table 1.

GNN terminology	Logical construct	Type of node	Notation
Input data	Ground fact h	Fact node	$\mathrm{F}_{(\mathrm{h},\vec{w})}$
Convolution	Ground rule's $\alpha\theta$ body	Rule node	$\mathrm{R}^{\mathrm{c}\theta}_{\alpha\theta}$
Pooling	Rule's α ground head h	Aggregation node	$G^{\mathrm{h}=\mathrm{c}\theta_{\mathrm{i}}}_{\alpha}$
Combination	Ground atom h	Atom node	A_{h}

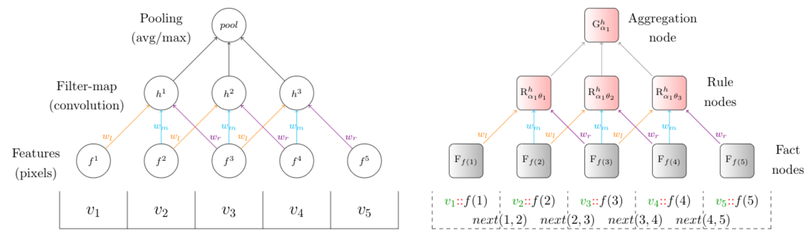
Table 1: Correspondence between the common GNN terminology and LRNN transformation of a logical ground (Herbrand) model into a (differentiable) computation graph.

Encoding Common Neural Architectures

Through the templating view, standard feed-forward NNs correspond with their expressiveness to *propositional* logic, which has been successfully exploited before [1]. With the *relational* logic, however, we can now move to the more interesting neural architectures.

Convolutional Neural Networks

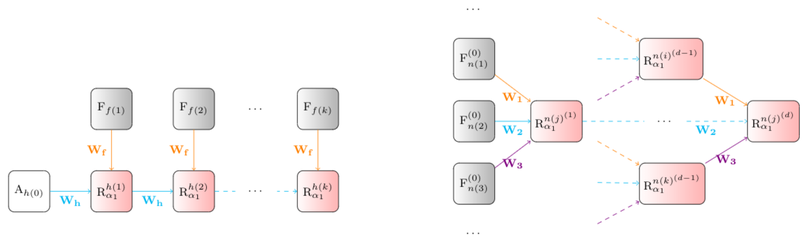
h :- w_{l}: f(A), w_{m}: f(B), w_{r}: f(C), next(A,B), next(B,C).



Recurrent and Recursive Networks

$\mathrm{h}(\mathrm{Y})$:- $\mathbf{W}_{\mathrm{f}}$:f(Y), $\mathbf{W}_{\mathrm{h}}$:h(X), next(X,Y).

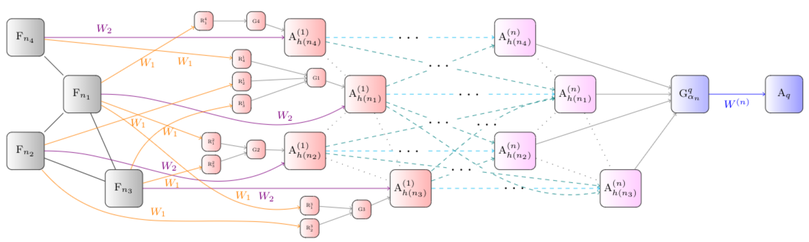
$\mathrm{n}(\mathrm{P})$:- $\mathbf{W}_1$:n(C_1), $\mathbf{W}_2$:n(C_3), $\mathbf{W}_3$:n(C_3), parent(P,C_1,C_2,C_3).



Graph Neural Networks

$\mathrm{h}^{(\mathrm{i})}(\mathrm{V})$:- $\mathbf{W}_1^{(\mathrm{i})}$: $\mathrm{h}^{(\mathrm{i}-1)}(\mathrm{U})$, edge(V,U).

$\mathrm{h}^{(\mathrm{i})}(\mathrm{V})$:- $\mathbf{W}_2^{(\mathrm{i})}$: $\mathrm{h}^{(\mathrm{i}-1)}(\mathrm{V})$.



Beyond GNNs

Note that the example logical templates are an actual *code* that is not only highly expressive, but can also be run very efficiently. To prove that, we compared against top GNN frameworks through standard GNN models on a classic molecular prediction benchmark (Table 2).

Table 2: Training times *per epocha* across the different models and frameworks. Additionally, the startup model creation time (grounding) overhead of LRNNs is displayed.

model/engine	LRNNs (s)	PyG (s)	DGL (s)	LRNNs startup (s)
GCN	**0.25 ± 0.01**	3.24 ± 0.02	23.25 ± 1.94	35.2 ± 1.3
g-SAGE	**0.34 ± 0.01**	3.83 ± 0.04	24.23 ± 3.80	35.4 ± 1.8
GIN	**1.41 ± 0.10**	11.19 ± 0.06	52.04 ± 0.41	75.3 ± 3.2

Crucially, in contrast to the existing frameworks, the LRNN framework is not limited to the GNN models, and much more expressive constructs can be designed just as easily.

Example: molecular rings

For instance, consider a simple learning scheme with molecular rings, which are already beyond the GNN expressiveness. But with the relational logic, declaring a ring is simple:

$$\_\mathrm{ring}_6(\mathrm{A},\ldots,\mathrm{F}) \text{ :- } \mathrm{bond}(\mathrm{A},\mathrm{B}),\ \ldots,\ \mathrm{bond}(\mathrm{E},\mathrm{F}),\ \mathrm{bond}(\mathrm{F},\mathrm{A}).$$

and its distributed representation can then be easily aggregated:

$$\mathbf{W}_{\mathrm{r}} :: \mathrm{ring}_6^{(\mathrm{n})}(\mathrm{A},\ldots,\mathrm{F}) \text{ :- } \_\mathrm{ring}_6(\mathrm{A},\ldots,\mathrm{F}),\ \mathbf{W}_{\mathrm{a}} : \mathrm{atom}^{(\mathrm{n})}(\mathrm{A}),\ \ldots,\ \mathbf{W}_{\mathrm{f}} : \mathrm{atom}^{(\mathrm{n})}(\mathrm{F}).$$

and propagated:

$$\mathbf{W}_{\mathrm{a}'} :: \mathrm{atom}^{(\mathrm{n})}(\mathrm{A}) \text{ :- } \mathbf{W}_{\mathrm{r}'} : \mathrm{ring}_6^{(\mathrm{n}-1)}(\mathrm{A},\ldots,\mathrm{F}).$$

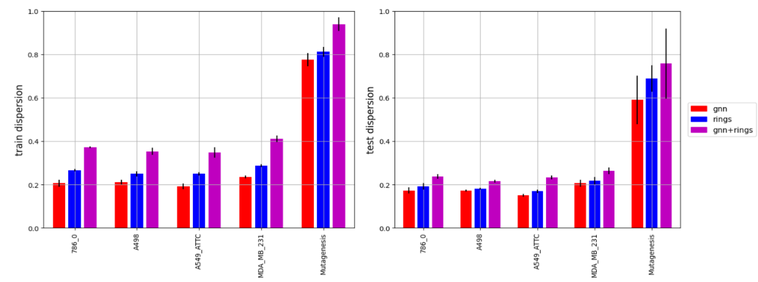


Figure 2: Comparison of the `gnn`, `rings`, and the joint `gnn + rings` learners across several molecule classification datasets w.r.t. training (left) and testing (right) dispersion.

References

Other examples of differentiable relational logic constructs beyond the GNN expressiveness are then described in the paper [3]. Please find the framework at github.com/GustikS/NeuraLogic with a recent Python frontend at pyneuralogic.readthedocs.io

We hope you will find it useful for design of your own deep *relational* learning ideas!

[1] Towell, G. G., Shavlik, J. W., and Noordewier, M. O. (1990) Refinement of approximate domain theories by knowledge-based neural networks. In *Proceedings of the eighth National conference on Artificial intelligence*.

[2] Sourek, G., Aschenbrenner, V., Zelezny, F., Schockaert, S., and Kuzelka, O. (2018) Lifted relational neural networks: Efficient learning of latent relational structures. *Journal of Artificial Intelligence Research,*.

[3] Šourek, G., Železný, F., and Kuželka, O. (2021) Beyond graph neural networks with lifted relational neural networks. *Machine Learning*, pp. 1–44.

*We acknowledge support by Czech Science Foundation grants 20-19104Y and 20-29260S.