Lossless Compression of Structured Convolutional Models (GNNs) via Lifting

Gustav Sourek* & Filip Zelezny & Ondrej Kuzelka

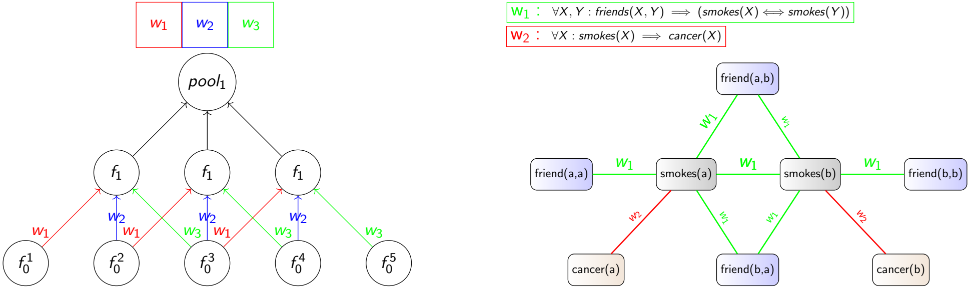
* *souregus@fel.cvut.cz*

Czech Technical University in Prague

@ *International Conference on Learning Representations (ICLR) '21*

Lifted (Templated) Models

Lifted (templated) models exploit input sample **symmetries** using parameterized computation **templates**, while inducing **weight-sharing** in their respective computation graphs:



Templating in a CNN (left) and a lifted graphical model (MLN - right).

Templating improves **generalization** w.r.t. different sample transformations in pixel grids (CNNs), trees (RNNs), graphs (GNNs), and beyond (Lifted Relational Neural Networks).

Lifted Inference

Lifted Inference

is a technique to exploit **computation graph symmetries** for **speedup**.

- known in *Statistical Relational Learning* and database query engines
- but so far *unexploited in deep learning*!

Q: But do such computation graph symmetries occur in deep learning practice?

- A: Yes, particularly in the recently explored **structured data** domains with **discrete-valued** elements such as social, biological, chemical graphs and networks.

"Structured Convolutional Models?"

are simply the templated (convolutional) models applied to such structured data domains.

- e.g. the common **Graph Neural Networks** (GNNs)

Lossless Compression via Lifting

To detect the symmetries in computation graphs, we partition its nodes into **equivalence classes**:

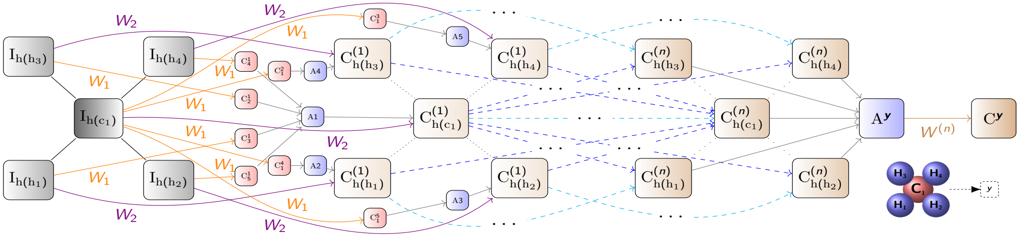
Definition (functional equivalence)

We say that two nodes $N_1, N_2 \in \mathcal{N}$ are **functionally equivalent** if, for any initialization of (shared) weights $\mathcal{W}$, it holds that $value(N_1; \mathcal{W}) = value(N_2; \mathcal{W})$.

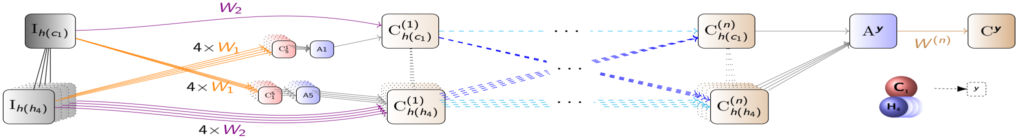
The compression algorithm then works very simply in 2 steps

1. precalculate the equivalence classes of neurons
2. iterate the computation graph and replace all nodes with a single equivalent

GNN compression example



is compressed into **equivalent** GNN computation graph:



Experimental settings

Models:

- classic GCNs, graph-SAGE, GIN, and some extensions (graphlet networks)
- also GNNs modified into knowledge-base embedding (KBE) mode

Learning tasks:

- *Molecule classification*
  - various (large) molecular datasets
- *Knowledge base completion* (KBC)
  - various mid-sized KBs such as Kinships, Nations, and UMLS

Lossless Compression Results

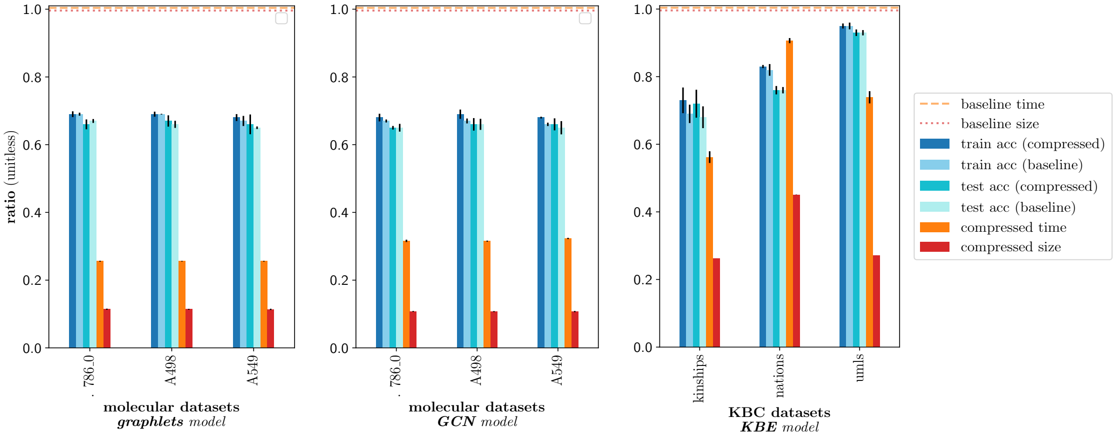
The models were implemented within the framework of *Lifted Relational Neural Networks*[1] (LRNNs), but we also compared with specialized GNN frameworks of PyG[2] and DGL [3]:

Model	Lifting (s)	LRNNs (s)	PyG (s)	DGL (s)
GCN	**0.25** ± **0.01**	0.75± 0.01	3.24 ± 0.02	23.25 ± 1.94
g-SAGE	**0.34** ± **0.01**	0.89± 0.01	3.83 ± 0.04	24.23 ± 3.80
GIN	**1.41** ± **0.10**	2.84± 0.09	11.19 ± 0.06	52.04 ± 0.41

Training times *per epocha* across different models and frameworks over 3000 molecules.

[1] https://github.com/GustikS/NeuraLogic
[2] https://github.com/rusty1s/pytorch_geometric
[3] https://www.dgl.ai/

Lossless Compression Results



Comparison of 3 different baseline models of graphlets (left), GCNs (middle), and KBEs (right) with their compressed versions over molecule classification (left, middle) and knowledge-base completion (right).

Conclusions

- The **symmetries are indeed common** in structured (discrete) data domains
  - especially in the molecular graphs, but also elsewhere
- The resulting compression rates and speedups are **quite significant**
  - easily 2-5x improvement in classic GNN classification settings
- The technique is **very simple** and can be adopted by any templated model
  - and can be easily extended into a lossy compression setting for further speedup
- The experiments and the LRNN framework itself are available at Github
  - Experiments: `https://github.com/GustikS/NeuraLifting`
  - LRNN: `https://github.com/GustikS/NeuraLogic`
  - email: `souregus@fel.cvut.cz`