Probabilistic Inference

reduces to

Weighted Model Counting (WMC)

Deep Learning

Intractable (#P-hard)

"Efficient" Approximation (in special cases)

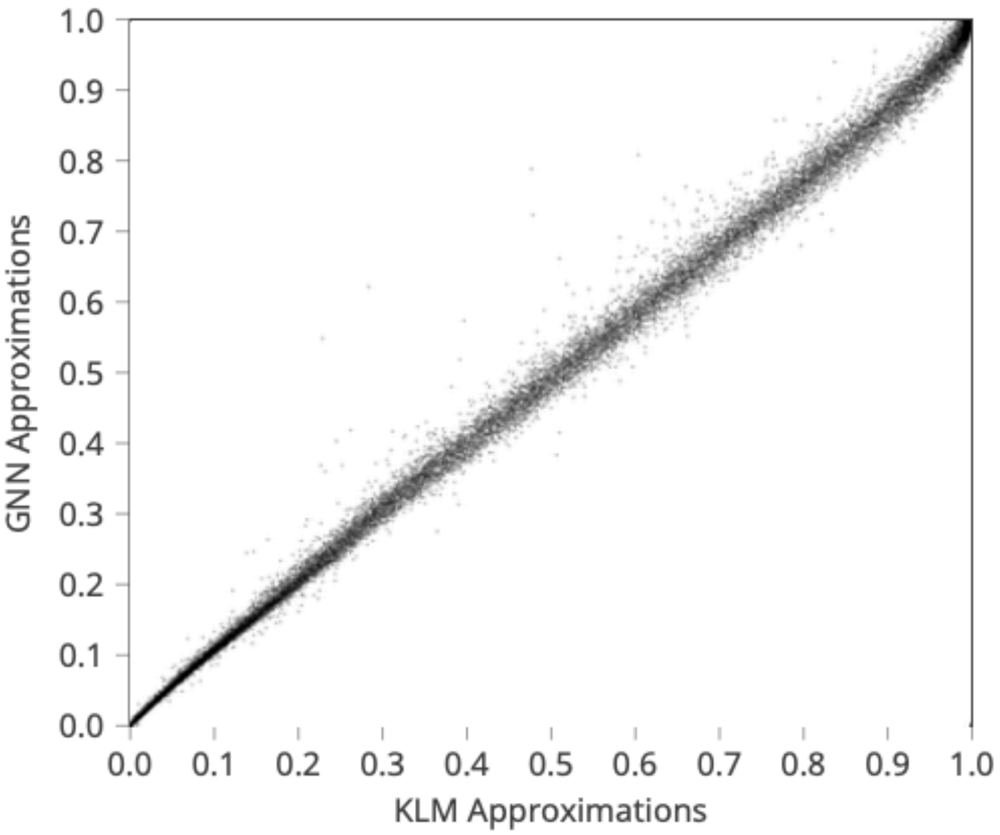
Breakthrough performance

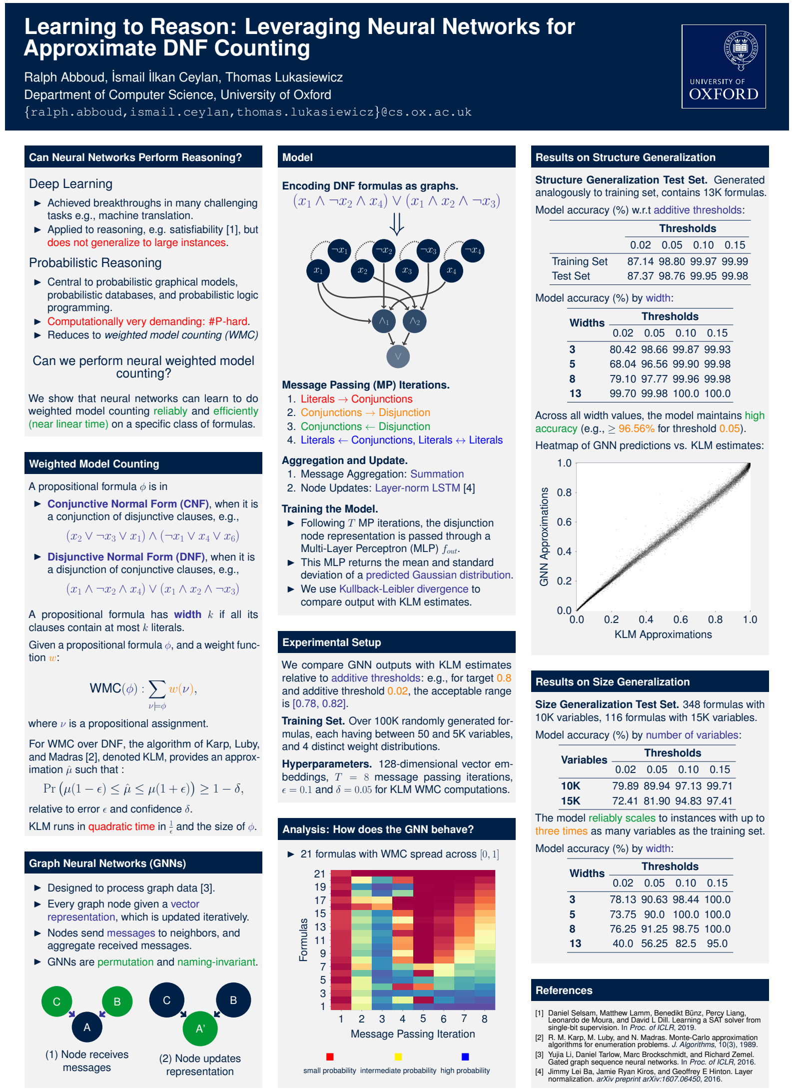
Reasoning Tasks: Poor Generalization

Neural WMC

Learning To Reason: Leveraging Neural Networks for Approximate DNF Counting

DLGMA Paper 20





Learning to Reason: Leveraging Neural Networks for Approximate DNF Counting

Ralph Abboud, İsmail İlkan Ceylan, Thomas Lukasiewicz
Department of Computer Science, University of Oxford
{ralph.abboud,ismail.ceylan,thomas.lukasiewicz}@cs.ox.ac.uk

UNIVERSITY OF OXFORD

Can Neural Networks Perform Reasoning?

Deep Learning

- Achieved breakthroughs in many challenging tasks e.g., machine translation.
- Applied to reasoning, e.g. satisfiability [1], but does not generalize to large instances.

Probabilistic Reasoning

- Central to probabilistic graphical models, probabilistic databases, and probabilistic logic programming.
- Computationally very demanding: #P-hard.
- Reduces to *weighted model counting (WMC)*

Can we perform neural weighted model counting?

We show that neural networks can learn to do weighted model counting reliably and efficiently (near linear time) on a specific class of formulas.

Weighted Model Counting

A propositional formula ϕ is in

- **Conjunctive Normal Form (CNF)**, when it is a conjunction of disjunctive clauses, e.g.,
$$(x_2 \vee \neg x_3 \vee x_1) \wedge (\neg x_1 \vee x_4 \vee x_6)$$
- **Disjunctive Normal Form (DNF)**, when it is a disjunction of conjunctive clauses, e.g.,
$$(x_1 \wedge \neg x_2 \wedge x_4) \vee (x_1 \wedge x_2 \wedge \neg x_3)$$

A propositional formula has **width** k if all its clauses contain at most k literals.

Given a propositional formula ϕ, and a weight function w:

$$\mathrm{WMC}(\phi) : \sum_{\nu \models \phi} w(\nu),$$

where ν is a propositional assignment.

For WMC over DNF, the algorithm of Karp, Luby, and Madras [2], denoted KLM, provides an approximation $\hat{\mu}$ such that :

$$\Pr\left(\mu(1-\epsilon) \leq \hat{\mu} \leq \mu(1+\epsilon)\right) \geq 1-\delta,$$

relative to error ϵ and confidence δ.

KLM runs in quadratic time in $\frac{1}{\epsilon}$ and the size of ϕ.

Graph Neural Networks (GNNs)

- Designed to process graph data [3].
- Every graph node given a vector representation, which is updated iteratively.
- Nodes send messages to neighbors, and aggregate received messages.
- GNNs are permutation and naming-invariant.

C B A
(1) Node receives messages

C B A'
(2) Node updates representation

Model

Encoding DNF formulas as graphs.

$$(x_1 \wedge \neg x_2 \wedge x_4) \vee (x_1 \wedge x_2 \wedge \neg x_3)$$

$\Downarrow$

x_1 $\neg x_1$ x_2 $\neg x_2$ x_3 $\neg x_3$ x_4 $\neg x_4$ $\wedge_1$ $\wedge_2$ $\vee$

Message Passing (MP) Iterations.

1. Literals → Conjunctions
2. Conjunctions → Disjunction
3. Conjunctions ← Disjunction
4. Literals ← Conjunctions, Literals ↔ Literals

Aggregation and Update.

1. Message Aggregation: Summation
2. Node Updates: Layer-norm LSTM [4]

Training the Model.

- Following T MP iterations, the disjunction node representation is passed through a Multi-Layer Perceptron (MLP) f_{out}.
- This MLP returns the mean and standard deviation of a predicted Gaussian distribution.
- We use Kullback-Leibler divergence to compare output with KLM estimates.

Experimental Setup

We compare GNN outputs with KLM estimates relative to additive thresholds: e.g., for target 0.8 and additive threshold 0.02, the acceptable range is [0.78, 0.82].

Training Set. Over 100K randomly generated formulas, each having between 50 and 5K variables, and 4 distinct weight distributions.

Hyperparameters. 128-dimensional vector embeddings, $T = 8$ message passing iterations, $\epsilon = 0.1$ and $\delta = 0.05$ for KLM WMC computations.

Analysis: How does the GNN behave?

- 21 formulas with WMC spread across $[0, 1]$

Formulas
21 19 17 15 13 11 9 7 5 3 1
1 2 3 4 5 6 7 8
Message Passing Iteration
small probability intermediate probability high probability

Results on Structure Generalization

Structure Generalization Test Set. Generated analogously to training set, contains 13K formulas.

Model accuracy (%) w.r.t additive thresholds:

	Thresholds			
	0.02	0.05	0.10	0.15
Training Set	87.14	98.80	99.97	99.99
Test Set	87.37	98.76	99.95	99.98

Model accuracy (%) by width:

Widths	Thresholds			
	0.02	0.05	0.10	0.15
3	80.42	98.66	99.87	99.93
5	68.04	96.56	99.90	99.98
8	79.10	97.77	99.96	99.98
13	99.70	99.98	100.0	100.0

Across all width values, the model maintains high accuracy (e.g., $\geq$ 96.56% for threshold 0.05).

Heatmap of GNN predictions vs. KLM estimates:

GNN Approximations
KLM Approximations
0.0 0.2 0.4 0.6 0.8 1.0

Results on Size Generalization

Size Generalization Test Set. 348 formulas with 10K variables, 116 formulas with 15K variables.

Model accuracy (%) by number of variables:

Variables	Thresholds			
	0.02	0.05	0.10	0.15
10K	79.89	89.94	97.13	99.71
15K	72.41	81.90	94.83	97.41

The model reliably scales to instances with up to three times as many variables as the training set.

Model accuracy (%) by width:

Widths	Thresholds			
	0.02	0.05	0.10	0.15
3	78.13	90.63	98.44	100.0
5	73.75	90.0	100.0	100.0
8	76.25	91.25	98.75	100.0
13	40.0	56.25	82.5	95.0

References

[1] Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L Dill. Learning a SAT solver from single-bit supervision. In *Proc. of ICLR*, 2019.

[2] R. M. Karp, M. Luby, and N. Madras. Monte-Carlo approximation algorithms for enumeration problems. *J. Algorithms*, 10(3), 1989.

[3] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. In *Proc. of ICLR*, 2016.

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.