ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1.
Available from http://www.i6doc.com/en/.

Graph Neural Networks for Propositional Model Counting

Gaia Saveri[1,2] and Luca Bortolussi[2]

1- University of Pisa - Department of Computer Science
Largo B. Pontecorvo 3, 56127 Pisa - Italy

2- University of Trieste - Department of Mathematics and Geoscience
Via A. Valerio 12, 34127 Trieste - Italy

Abstract. Graph Neural Networks (GNNs) have been recently leveraged to solve several logical reasoning tasks. Nevertheless, counting problems such as propositional model counting (#SAT) are still mostly approached with traditional solvers. Here we tackle this gap by presenting an architecture based on the GNN framework for belief propagation (BP) of [1], extended with self-attentive GNN and trained to approximately solve the #SAT problem. We experimentally show that our model, trained on a small set of random Boolean formulae, is able to scale effectively to much larger problem sizes, outperforming state of the art approximate solvers. Moreover, we show that it can be efficiently fine-tuned to provide good generalization results on different formulae distributions, such as those coming from SAT-encoded combinatorial problems.

1 Introduction

Propositional model counting (#SAT), the task of computing the number of satisfying assignments of a Boolean formula, is of relevant importance in computer science as it arises in many domains, such as Bayesian reasoning and combinatorial designs. Nevertheless, #SAT has been proven to be #P-complete, thus computationally at least as hard as NP-complete problems [2]. For this reason, state-of-the-art exact #SAT solvers are not capable of handling industrial-size problems, and a number of approximate solvers have been developed. On the other hand, there is an increasing interest in leveraging machine learning to solve logical and combinatorial reasoning tasks [3]. Graph Neural Networks (GNNs) [4] fit well in this scenario as they carry inductive biases that effectively encode combinatorial inputs, such as permutation invariance and sparsity awareness. In this work we investigate whether GNNs can be meaningfully applied to approximately solve the #SAT problem, an objective that, to the best of our knowledge, is only tackled in [1]. To this end, we extend the architecture presented in [1], by augmenting the model with a self-attention mechanism. We experimentally show (code is available at: `https://github.com/GaiaSaveri/GNN-sharpSAT`) that our model, trained on a small set of random Boolean formulae, is able to scale-up to larger problem sizes, outperforming state-of-the art approximate #SAT solver. Moreover we describe a simple yet effective fine-tuning strategy, that allows the model to generalize across diverse data distributions, with only a few tens of labeled formulae required.

Related Work. In [5] a graph neural network model is proposed to solve the weighted disjunctive normal form counting problem (weighted #DNF). Moreover, a significant

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

body of work has been developed to learn how to solve NP-complete problems leveraging GNNs [3]. Among them, methods tackling the Boolean satisfiability problem (SAT) are of interest for this work. In particular [6] is the most similar in spirit to our method, as it leverages a GNN as end-to-end SAT solver.

2 Background

Belief Propagation and #SAT. Belief Propagation (BP) [7] is an approximate inference algorithm for computing marginals of a probability distribution, exploiting the factor graph arising from its factorization. Considering a discrete probability distribution over variables $V = \{x_1, \ldots, x_n\}$ (below $x_{\mathcal{N}(f_j)}$ is the set of variables each factor f_j depends on):

$$P(x_1, \ldots, x_n) = \frac{1}{Z} \prod_{j=1}^{m} f_j(x_{\mathcal{N}(f_j)}),\ Z = \sum_{x \in V} \prod_{j=1}^{m} f_j(x_{\mathcal{N}(f_j)}) \tag{1}$$

then BP computes an approximation of the factor and variable marginals (also called beliefs) by iteratively passing messages between neighboring nodes on the factor graph. Beliefs can be used to compute a variational approximation of the partition function of the factor graph Z [8]. Propositional formulae, that w.l.o.g. we assume to be in Conjunctive Normal Form (CNF), can be translated into a factor graph containing a factor node for every clause and a variable node for every variable in the formula, and undirected edges connecting variable nodes to the factor nodes of the clauses they appear in. If we impose that a factor node takes value 1 for variable configurations that satisfy the corresponding clause and 0 otherwise, then the partition function of this factor graph (Z of Equation 1) counts, by construction, the number of models of the input formula (this also allows us to distinguish between two formulae differing only for literals' negation, which in principle have the same factor graph representation). Such intuition enables the adoption of probabilistic reasoning methods as approximate #SAT solvers [9].

Graph Attention Networks (GATs). GATs [10] are a type of GNNs endowed with a self-attention mechanism, that allows the network to aggregate the information coming from different nodes of the input graph putting a different weight on some entities, and fade out the rest. This is done computing attention coefficients for all pairs of neighboring nodes, and updating each node's representation by a weighted combination of the node embeddings and the above mentioned attention coefficients. To stabilize the learning process, a multi-head attention is applied by replicating the operations of the layer independently K times and aggregating the results with a feature-wise aggregation function such as concatenation, sum or average.

3 Our approach: GAT for #SAT

The objective of this work is to tackle the #SAT problem using a GNN model as an approximate solver. Our starting point is (one of the variants of) the architecture proposed in [1]. This model, called Belief Propagation Neural Network (BPNN), generalizes BP by means of GNNs, taking as input the factor graph representing a CNF

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

SAT formula and giving as output an estimation of the logarithm of the factor graph's partition function Z of Equation 1. BPNN keeps the algorithmic structure of BP and augments its message passing scheme by transforming, at each iteration, the messages exchanged at the previous iteration using MLPs, before aggregating and sending them to the neighboring nodes. Once the message passing phase is completed, a readout phase is executed, which outputs an estimation $\ln \hat{Z}$ of the natural logarithm of the partition function Z of the input factor graph, leveraging the network's approximation of variable and factor beliefs, computed as in standard BP.

3.1 Neural Belief Propagation with Attention

GNNs endowed with an attention mechanism are a promising research direction in the neuro-symbolic computing field, as they might enhance structured reasoning and efficient learning [3]. This is why we augment the BPNN architecture with a GAT-style attention mechanism. We will refer to our architecture as BPGAT, as it puts together the algorithmic structure of BP and the computational formalism of GATs. In BPGAT both factor-to-variable ($\hat{m}_{j\to i}$) and variable-to-factor ($\hat{m}_{i\to j}$) messages are aggregated using a GAT-style multi-head attention mechanism. At iteration $k+1$, for each variable-to-factor message $\hat{m}_{i\to j}$, attention coefficients α_{ij} for every factor node f_j in the neighborhood of a variable node x_i are computed as in [10] (with $W \in \mathbb{R}^{d\times d}$ being learnable matrix and $\mathbf{a} \in \mathbb{R}^{2d}$ the set of parameters of a single-layer feedforward neural network):

$$\alpha_{ij}^{(k+1)} = \frac{\exp(\text{LeakyReLU}(\mathbf{a}^T[\text{concat}(W\hat{m}_{i\to j}^{(k)}, W\hat{m}_{j\to i}^{(k)})]))}{\sum_{k\in\mathcal{N}(x_i)} \exp(\text{LeakyReLU}(\mathbf{a}^T[\text{concat}(W\hat{m}_{i\to j}^{(k)}, W\hat{m}_{k\to i}^{(k)})]))} \tag{2}$$

Analogous computations are performed to obtain the attention coefficients for factor-to-variable messages. Once the aggregation of the incoming messages is performed, the new messages are computed as in standard BP. After T iterations, the same readout phase of BPNN is executed, in order to obtain $\ln \hat{Z}$, i.e. an approximation of the natural logarithm of the number of models of the input CNF SAT formula.

4 Experimental Evaluation

4.1 Experimental Setting

The model is trained for 1000 epochs to minimize the Mean Squared Error (MSE) between the natural logarithm $\ln Z$ of the true number of models of the input formula, and the output of the model $\ln \hat{Z}$. In order to compute attention coefficients, we used a 3-layer GAT network, having respectively $4, 4, 6$ attention heads. For comparison, we also implemented the BPNN model, using the parameters detailed in [1]. The number of iterations T of the message passing scheme has been set to 5 [1].

The training dataset $\mathcal{D} = \{(\phi_i, \ln Z_i)\}$ consists of a set of 1000 pairs of CNF SAT formulae ϕ_i and the logarithm $\ln Z_i$ of their true model count. Such formulae

[1] These choices have been made after a set of preliminary and ablation studies, whose results can be found in an extended version of our paper (containing also a mathematical formulation of the model), available online at `https://arxiv.org/abs/2205.04423`.

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

are drawn from a distribution of random formulae built by uniformly sampling, for every formula, the number of variables and the number of clauses (given the input ranges (nv_{min}, nv_{max}) and (nc_{min}, nc_{max}), respectively). For each clause, a number $k \sim 2 + \text{Bernoulli}(0.7) + \text{Geometric}(0.4)$ of variables are chosen, so that each clause has 5 variables in average. The number of models for each generated formula is obtained with the exact #SAT solver sharpSAT [11]. The dataset used for training both BPGAT and BPNN has been generated using $(nv_{min}, nv_{max}) = (10, 30)$ and $(nc_{min}, nc_{max}) = (20, 50)$; this produced formulae having an average of 19.87 variables and 34.87 clauses. This is a relevant difference w.r.t. [1] in which training data consist in a sample of formulae drawn from the same benchmarks used to test the architecture (which might be more costly to obtain).

Fine-Tuning Protocol. To allow the model to extrapolate to data distributions different from the one seen during training, without requiring a large amount of distribution-specific labeled data, as weight initializer for fine-tuning towards new distributions we use BPGAT trained for 500 epochs with the protocol and data described earlier in this Section, fine-tuning it for additional 250 epochs, with only 250 labeled examples.

4.2 Results

The objective of our experiments is twofold: evaluating both BPGAT scalability and generalization. As a baseline, we used ApproxMC [12], the state-of-the-art approximate #SAT solver, which is a randomized hashing algorithm that provides Probably Approximately Correct (PAC) guarantees.[2]

Scalability. In order to assess the scalability of our model, i.e. the ability to perform well on formulae which are from the same data generating distribution than the ones seen during training but of larger size, we generated several datasets following the procedure detailed in Section 4.1. Table 1 shows the statistics of the datasets used in this testing phase, each containing 300 labeled instances. It is worth noting that all datasets contain much larger formulae sizes than the one seen during training. Table 1 shows the results obtained, in terms of Root Mean Square Error (RMSE) and Mean Relative Error (MRE) between the true $\ln Z$ and the output of the model $\ln \hat{Z}$, by BPGAT, BPNN and ApproxMC. Remarkably, for all the datasets tested, BPGAT outperforms ApproxMC in terms of MRE (although not in terms of RMSE). Such higher RMSE is a consequence of few outliers with a large prediction error for BPGAT, while most of its predictions are close to the ground truth labels, as certified by the consistently lower MRE.

Out-of-distribution generalization. The second set of tests we performed aims at evaluating the generalization capabilities of our model. The problem classes we perform experiments with are both SAT-encoded combinatorial problems (k-dominating set, graph k-coloring, k-clique detection) and network QMR (Quick Medical Reference) problems taken from the test suite of [12]. As before, sharpSAT [11] is used to obtain

[2]Comparison with other guarantee-less counters such as [13] has been hindered by the lack of available implementations, to the best of our knowledge.

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

Dataset	Avg#var	Avg#cl	BPNN	BPGAT	ApproxMC
Test 1	61.8	76.89	0.3140/0.004072	0.1276/**0.001366**	**0.002025**/0.001576
Test 2	60.43	143.61	1.3623/0.01786	0.3100/**0.003201**	**0.2262**/0.02003
Test 3	124.07	75.26	0.3046/0.004131	0.1748/**0.001471**	**0.1035**/0.01134
Test 4	377.59	275.11	2.9112/0.01681	1.2061/**0.007433**	**0.4275**/0.04038

Table 1: Average number of variables, average number of clauses, of the datasets used to test scalability. RMSE/MRE performance of BPNN, BPGAT and ApproxMC on the resp. datasets.

Dataset	FT_BPGAT	TS_BPGAT	RF_BPGAT	ApproxMC	FT_BPNN
Network	0.2580/**0.005271**	1.9334/0.04887	14.2839/0.3608	**0.07619**/0.05403	0.3187/0.007469
Domset	0.5508/0.04252	1.7808/0.7125	11.9190/9.5070	**0.08155/0.02856**	1.0646/0.08392
Color	1.2110/0.1774	1.3430/0.2046	26.1898/5.9593	**0.09426/0.04241**	2.9254/0.8046
Clique	**0.007834/0.002475**	0.01773/0.007333	1.9625/0.8983	0.01113/0.04795	0.01201/0.004069

Table 2: RMSE/MRE performance of BPGAT and BPNN when fine-tuned for the specific data distribution (FT_BPGAT and FT_BPNN, resp.) following the protocol described in Section 4.1, when trained on the specific data distribution for 500 epochs, with 250 labeled examples (TS_BPGAT) and when trained on random formulae (RF_BPGAT) with the training protocol and the data generating procedures detailed in Section 4.1 and ApproxMC.

the number of models for each formula. Statistics of the datasets used to test generalization are as follows: formulae encoded from the $k-$dominating set, the k-coloring and the k-clique detection problem have an average of $38.43, 65.63, 46.37$ variables and $510.15, 294.54, 1145.73$ clauses, respectively; network QMR problems have a mean of 113 variables and 294.7 clauses. Each of the test set contains 300 previously unseen formulae. To asses the effectiveness of our fine-tuning protocol we made several experiments, whose results are summarized in Table 2. Interestingly, fine-tuning a model pre-trained on small random Boolean formulae (FT_BPGAT) gives better results than the same model trained on the specific dataset (TS_BPGAT): this is relevant in terms of efficiency of the architecture, as it requires only 250 labeled distribution-specific samples and 250 additional training epochs. Results of the comparison between our fine-tuned model (FT_BPGAT) and ApproxMC are shown in Table 2. It is worth observing that the performance of our architecture is comparable and in some cases outperforming that of ApproxMC (especially in terms of MRE). For completeness, we report also the performance of BPNN, fine-tuned with the same procedure as BPGAT. Overall, we observe that the architectural improvements guarantee a sensibly better performance in terms of scalability and generalizability over the BPNN model, likely because the attentional layer allows the network to focus (i.e. to give more weight) on regions of the input formulae which are more significant for the #SAT problem.

Data and Time-efficiency. The BPGAT architecture can be claimed data-efficient, as it requires only 1000 CNF small random formulae for (pre)training. Generating such training set requires $\sim 5s$ with the procedure described in Section 4.1. About time efficiency, once trained the BPGAT architecture is able to process (independently on the size of the formulae), all test instances described in this work, in a maximum of $3s$,

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

without leveraging GPU acceleration. This is much less than the time needed by the exact solver sharpSAT (which takes on average $286s$ for an instance of the dataset Test 4 described in Section 4.1) and by the approximate solver ApproxMC with standard parameters (which takes on average $200s$ for an instance of the same dataset).

5 Conclusions and Future Work

We presented BPGAT, an extension of the BPNN architecture presented in [1], which combines the algorithmic structure of belief propagation and the learning paradigm of graph attention networks. We conducted several experiments to investigate the scalability and generalization abilities of our network, showing that it is able to achieve a performance comparable to (and in some settings higher than) state-of-the-art approximate #SAT solvers, albeit the lack of any theoretical guarantees on the quality of the solution. Finally, we highlighted the efficiency of our model, both in terms of required training data and of running time. As future research directions, we will analyze the viability of extending BPGAT to tackle weighted conjunctive normal form model counting (weighted #CNF) problems. In this scenario, a straightforward application of our model would be that of approximate probabilistic inference on Bayesian Networks, which in many cases (e.g. when solved using variational inference) comes without any statistical guarantees.

References

[1] J. Kuck, S. Chakraborty, H. Tang, R. Luo, J. Song, A. Sabharwal, and S. Ermon. Belief Propagation Neural Networks. In *NeurIPS*, 2020.

[2] L. G. Valiant. The Complexity of Enumeration and Reliability Problems. *SIAM J. Comput.*, 8(3):410–421, 1979. Publisher: Society for Industrial and Applied Mathematics.

[3] Y. Bengio, A. Lodi, and A. Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. *Eur. J. Oper. Res.*, 290(2):405–421, 2021.

[4] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The Graph Neural Network Model. *IEEE Trans. Neural Networks*, 20(1):61–80, 2009.

[5] R. Abboud, İ. İ. Ceylan, and T. Lukasiewicz. Learning to Reason: Leveraging neural networks for approximate DNF counting. In *AAAI*, 2020.

[6] D. Selsam, M. Lamm, B. Bunz, P. Liang, L. de Moura, and D. L. Dill. Learning a SAT Solver from Single-Bit Supervision. In *ICLR*, 2019.

[7] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[8] H. A. Bethe. Statistical Theory of Superlattices. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 150(871):552–575, 1935.

[9] L. Kroc, A. Sabharwal, and B. Selman. Leveraging belief propagation, backtrack search, and statistics for model counting. *Ann. Oper. Res.*, 184(1):209–231, 2011.

[10] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph Attention Networks. In *ICLR*, 2018.

[11] M. Thurley. sharpSAT - Counting Models with Advanced Component Caching and Implicit BCP. In *SAT 2006*, volume 4121 of *LNCS*, pages 424–429. Springer, 2006.

[12] M. Soos and K. S. Meel. Bird: Engineering an efficient cnf-xor sat solver and its applications to approximate model counting. In *AAAI*, 2019.

[13] W. Wei and B. Selman. A new approach to model counting. In *SAT*, 2005.