AUGMENT WITH CARE: CONTRASTIVE LEARNING FOR THE BOOLEAN SATISFIABILITY PROBLEM

Anonymous authors
Paper under double-blind review

ABSTRACT

Supervised learning can improve the design of state-of-the-art solvers for combinatorial problems, but labelling large numbers of combinatorial instances is often impractical due to exponential worst-case complexity. Inspired by the recent success of contrastive pre-training for images, we conduct a scientific study of the effect of augmentation design on contrastive pre-training for the Boolean satisfiability problem. While typical graph contrastive pre-training uses label-agnostic augmentations, our key insight is that many combinatorial problems have well-studied invariances, which allow for the design of *label-preserving augmentations*. We find that label-preserving augmentations are critical for the success of contrastive pre-training. We show that our representations are able to achieve comparable test accuracy to fully-supervised learning while using only 1% of the labels. We also demonstrate that our representations are more transferable to larger problems from unseen domains.

1 INTRODUCTION

Combinatorial problems, e.g., Boolean satisfiability (SAT) or mixed-integer linear programming (MILP), have many applications in the industry and can encode many fundamental computational tasks. These problems are NP-complete, so solvers that perform efficiently in the worst case are not within reach. However, learning can be used to improve the average complexity of solvers on the population of combinatorial problems found in the wild (Nair et al., 2020).

Supervised learning is a promising approach to combinatorial solver design (e.g., Selsam & Bjørner, 2019; Nair et al., 2020). Unfortunately, the need for labels is a severe limitation. Many read-world problems, such as cryptography SAT instances, are extremely hard or, in some cases, even impossible to solve (Nejati & Ganesh, 2019). Computing expert branching labels for large-scale MILPs requires sophisticated parallel solvers (Nair et al., 2020).

In order to scale learning for combinatorial problems, we ask: how much can we learn from *unlabelled* combinatorial instances? In this work, we consider a contrastive learning approach, which begins by creating multiple "views" of every unlabelled instance, a process called augmentation. An encoder is trained to maximize the similarity between the representations of augmentations that come from the same instance, while minimizing the similarity between those of distinct ones (Chen et al., 2020a). This has been successful in computer vision: contrastive representations can be used with linear predictors to achieve competitive accuracies on ImageNet using a fraction of the labelled instances (Chen et al., 2020a; He et al., 2020; Chen et al., 2020b).

Our key insight is that combinatorial problems have well-studied invariances that can be used to design extremely effective augmentations for contrastive learning. Contrastive learning theory indicates that augmentations should be (roughly) label-preserving in order to confer guarantees on downstream prediction (Arora et al., 2019; Tosh et al., 2021; HaoChen et al., 2021). This is in contrast to the majority of label-agnostic graph contrastive frameworks (e.g., You et al., 2020; Hassani & Khasahmadi, 2020). For some combinatorial problems, label-preserving transformations are available from subroutines of existing solvers, e.g., variable elimination modifies SAT formulas while preserving their satisfiability. Our augmentations produce new formulas by randomly applying such satisfiability-preserving transformations. Crucially, our augmentations do not require full solves and are much cheaper to compute than the labels.

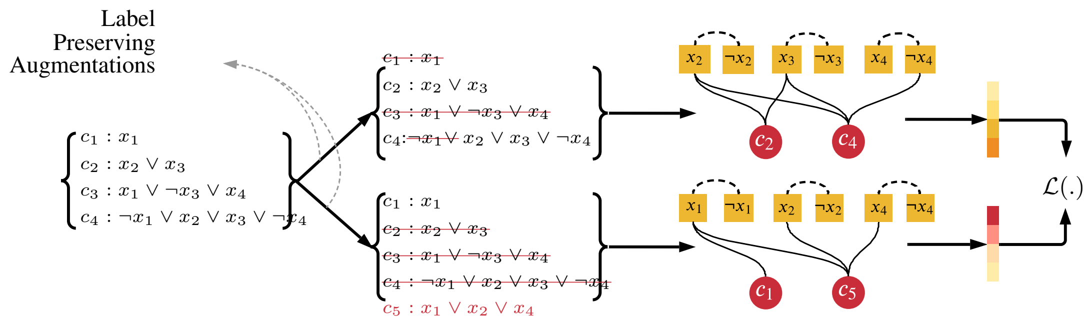
Label Preserving Augmentations
$c_1 : x_1$
$c_2 : x_2 \vee x_3$
$c_3 : x_1 \vee \neg x_3 \vee x_4$
$c_4 : \neg x_1 \vee x_2 \vee x_3 \vee \neg x_4$
$c_5 : x_1 \vee x_2 \vee x_4$
$\mathcal{L}(.)$

Figure 1: Our contrastive learning framework for combinatorial problems. Given an instance ϕ, a pair of augmented samples $(\hat{\phi}_1, \hat{\phi}_2)$ are formed using label-preserving augmentations. $(\hat{\phi}_1, \hat{\phi}_2)$ are then transformed to graph formats (G_1, G_2). An encoder is used to extract graph representations $(\mathbf{z}_1, \mathbf{z}_2)$. Lastly, a standard contrastive loss is applied over a mini-batch of instances.

We study data augmentations for contrastive learning for SAT. We demonstrate that:

- Augmentation design is critical for contrastive pre-training on combinatorial problems. In particular, our augmentations are sufficient for strong performance, while existing graph augmentations are not.
- Our contrastive method matches the best supervised baseline while using $100\times$ fewer training labels.

2 Background and Related Work

SAT. A Boolean formula in propositional logic consists of Boolean variables composed by logical operators "and" ($\wedge$), "or" ($\vee$) and "not" ($\neg$). A *literal* is a variable v or its negation $\neg v$. A *clause* is a disjunction of literals $\bigvee_{i=1}^{n} l_i$. *SAT* is the problem of deciding, for a given formula ϕ, if there exists a satisfying assignment (SAT) or not (UNSAT). We can represent a SAT formula ϕ by a bipartite graph called *literal-clause incidence graph* (LIG$^+$) (Figure 4 in Appendix B).

Graph Contrastive Learning. Existing graph contrastive frameworks can, in principle, be used to learn representations for combinatorial problems. Common augmentations in graph contrastive learning include: 1) perturbing structures, such as, node dropping, subgraph sampling or graph diffusion (You et al., 2020), and 2) perturbing features, such as, masking or adding noise to the node features (Hassani & Khasahmadi, 2020).

3 Methodology

3.1 Framework Overview

Similar to contrastive algorithms in other domains (e.g., images), we learn representations by contrasting augmented views of the same instance against negative samples. Inspired by SimCLR Chen et al. (2020a), we propose a contrastive learning framework for combinatorial problems (Figure 1). Our framework consists of four major components: augmentations, format transformation, encoder and contrastive loss. The detailed description of each component can be found in Appendix C.

3.2 Label-preserving Augmentations

To guarantee downstream predictive performance, augmentations for contrastive learning should (mostly) preserve the labels of downstream tasks (Arora et al., 2019; Tosh et al., 2021; HaoChen et al., 2021; Dubois et al., 2021). Label-preserving augmentations (LPAs) are well-studied for many combinatorial problems and are much cheaper to obtain than labels. For SAT, LPAs should preserve satisfiability of an instance. Common preprocessing techniques from SAT solvers can be used as LPAs. Our work studies four common LPAs for SAT: add unit literal (AU), subsumed clause

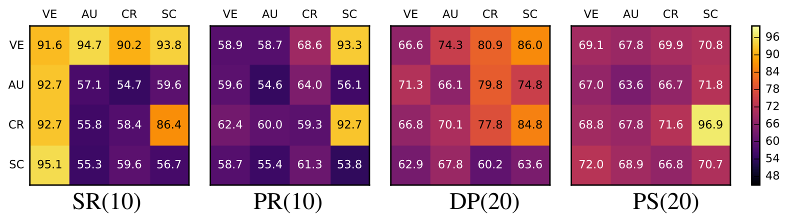

Figure 2: Heatmap: linear evaluation accuracy of SSL models trained with different single LPA augmentations (diagonal) or paired combinations (off-diagonal). Each column denotes different datasets. For off-diagonal entries on each heatmap, the row corresponds to the first augmentation applied.

elimination (SC), clause resolution (CR) and variable elimination (VE). Their detailed descriptions are in Appendix D.

4 Empirical Study of Augmentations

Our first experiments study the role of augmentations in contrastive learning for SAT prediction. Following the standard linear evaluation protocol to evaluate representations (Chen et al., 2020a), we report test accuracy of a linear classifier trained on top of frozen representations. We primarily used the encoder of NeuroSAT (Selsam et al., 2018) as the GNN architecture. We experimented using four SAT generators: SR (Selsam et al., 2018), Power Random 3SAT (PR) (Ansótegui et al., 2009), Double Power (DP) and Popularity Similarity (PS) (Giráldez-Cru & Levy, 2017). The generator produced a set of new unlabelled instances for each batch during SSL training. We used 100 labelled instances to train our linear evaluators. The test set consisted of 10^4 instances. More details of architectures, experiments and datasets are in Appendix E and Appendix H.

We also adopted four label-agnostic augmentations (LAAs) from a graph contrastive learning framework, GraphCL (You et al., 2020): *drop clauses* (DC), *drop variables* (DV), *link perturbation* (LP) and *subgraph* (SG). Unlike LPAs, these LAAs have no guarantee of preserving satisfiability. The heatmaps for LPAs and LAAs on four datasets are shown in Figure 2 and 5 respectively.

LPAs learn significantly better representations than LAAs. As Figure 2 and Figure 5 show, the accuracy $(\%)$ for the best LPA pair is: 95.1 for SR, 93.3 for PR, 86.0 for DP and 96.9 for PS, while the corresponding number $(\%)$ for LAAs is much lower: $54.4, 59.1, 70.0$ and 65.5. These results imply that SSL models trained with LPAs learn significantly better representations than those with LAAs. This performance gap between LPAs and LAAs meets our intuition. LAAs do not guarantee preserving satisfiability, which could result in false positive pairs that hurt the SAT prediction performance.

Resolution-based augmentations are the most powerful. In Figure 2, the best pair for each dataset includes either CR or VE. Both of them are based on the resolution rule in propositional logic. Resolution is a powerful inference rule in propositional logic. In fact, we can build a sound and complete propositional theorem prover with only resolutions (Genesereth & Kao, 2013). The Davis–Putnam algorithm (Davis & Putnam, 1960), the basis of practical SAT solvers, iteratively applies resolution until reaching satisfiability certificates. Resolution-based augmentations in contrastive learning may help the neural work learn the essence of resolution, which leads to better satisfiability prediction.

Composing different augmentations is beneficial across datasets. The highest accuracy in Figure 2 always comes from off-diagonal entries, i.e., composition of different augmentations. In PR, every singular augmentation failed to obtain accuracy higher than 60% by itself. While singular augmentations achieved decent accuracy on SR and DP, combinations further improved the accuracy. Similar to image and graph, composing different augmentations resulted in harder positives and better representations.

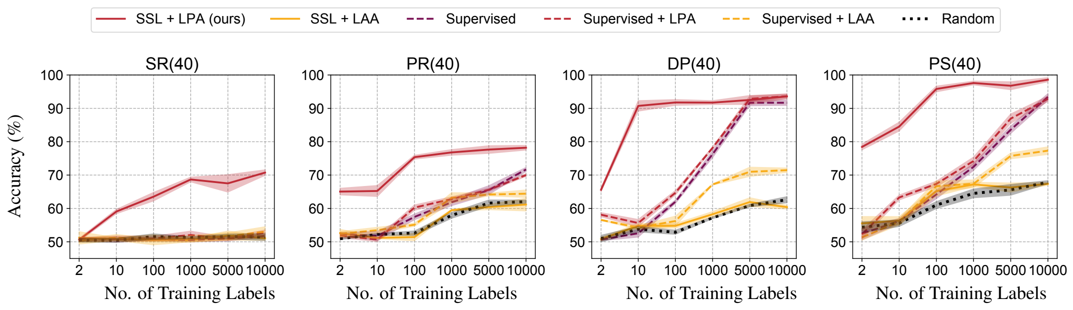

Figure 3: Our method (SSL + LPA) achieves significantly higher accuracy after linear evaluation and fine-tuning than all baselines in low-label regime and is comparable to supervised models that have been given more labels. We vary the number of training labelled instances from 2 to 10^4 and report the average accuracy and standard error over 3 trials for all methods.

5 Comparison with Other Methods

In addition to our model (SSL with LPA), we studied 4 baselines: SSL with LAA, supervised models trained without augmentations, supervised models trained with LPA or LAA. The details of experiments are in G. We varied the number of labelled instances from 2 to 10^4.

As shown in Figure 3, our method achieved substantially higher accuracy than others across all datasets in the low-label regime. For example, with 10 training labelled instances, the improvement $(\%)$ of ours compared from the second best method was: 9.42 for SR, 14.07 for PR, 30.81 for DP, 22.18 for PS. Our model's accuracy was also on par with supervised models that had access to significantly more labels: for all datasets, the accuracy of our models trained with 100 labels matched or exceeded the accuracy of our best supervised baselines trained with 10^4 labels, a $100\times$ reduction in the number of labels needed.

On the other hand, SSL models trained with LAAs were not much better than random-initialized ones under linear evaluation. We also found that using LAAs for supervised models even hurt performance, possibly because LAAs add label noise. For example, with 10000 labels on DP(40), adding LAAs gave 20.21% lower accuracy than supervised without augmentations.

6 Conclusions and Outlook

We studied the effect of data augmentations on contrastive learning for the Boolean satisfiability problem. We designed label-preserving augmentations using well-studied transformations, e.g., clause resolution or variable elimination, and confirmed the hypothesis that data augmentations should be label-preserving to help in downstream prediction. The design of our augmentations was critical; we found that resolution-based augmentations, which produced more distinct augmentations, were necessary for strong results. Our contrastive method was able to learn strong SAT predictors (at least as strong as our best supervised baselines) with $100\times$ fewer labelled training instances.

Although our results are restricted to Boolean satisfiability, they hold lessons for solver design more broadly. For example, the convex hull of MILP feasible sets is invariant to cuts, which suggests that contrastive pre-training could be used to improve Neural Diving (Nair et al., 2020). In general, our results strongly suggest that studying and exploiting invariances can dramatically improve the sample complexity of heuristics learned via imitation learning.

The study of combinatorial problems is fruitful for the broader machine learning community, because these problems are non-trivial and so much is known about their invariances. In particular, experiments can be designed that exactly satisfy the assumptions of invariant learning theory, making combinatorial problems fantastic test beds for the emerging field of contrastive and self-supervised learning.

REFERENCES

Amizadeh, S., Matusevych, S., and Weimer, M. Learning to solve circuit-sat: An unsupervised differentiable approach. In *ICLR*, 2019a.

Amizadeh, S., Matusevych, S., and Weimer, M. Pdp: A general neural framework for learning constraint satisfaction solvers. *arXiv preprint arXiv:1903.01969*, 2019b.

Ansótegui, C., Bonet, M. L., and Levy, J. Towards industrial-like random sat instances. In *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

Arora, S., Khandeparkar, H., Khodak, M., Plevrakis, O., and Saunshi, N. A theoretical analysis of contrastive unsupervised representation learning. *arXiv preprint arXiv:1902.09229*, 2019.

Bengio, Y., Lodi, A., and Prouvost, A. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.

Cappart, Q., Chételat, D., Khalil, E., Lodi, A., Morris, C., and Veličković, P. Combinatorial optimization and reasoning with graph neural networks. *arXiv preprint arXiv:2102.09544*, 2021.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020a.

Chen, X., Fan, H., Girshick, R., and He, K. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020b.

Dai, H., Khalil, E. B., Zhang, Y., Dilkina, B., and Song, L. Learning combinatorial optimization algorithms over graphs. *arXiv preprint arXiv:1704.01665*, 2017.

Davis, M. and Putnam, H. A computing procedure for quantification theory. *Journal of the ACM (JACM)*, 7(3):201–215, 1960.

Dubois, Y., Bloem-Reddy, B., Ullrich, K., and Maddison, C. J. Lossy compression for lossless prediction. In *NeurIPS*, 2021.

Eén, N. and Biere, A. Effective preprocessing in sat through variable and clause elimination. In *International conference on theory and applications of satisfiability testing*, pp. 61–75. Springer, 2005.

Gasse, M., Chételat, D., Ferroni, N., Charlin, L., and Lodi, A. Exact combinatorial optimization with graph convolutional neural networks. *arXiv preprint arXiv:1906.01629*, 2019.

Genesereth, M. and Kao, E. Introduction to logic. *Synthesis Lectures on Computer Science*, 4(1): 1–165, 2013.

Giráldez-Cru, J. and Levy, J. Locality in random sat instances. International Joint Conferences on Artificial Intelligence, 2017.

HaoChen, J. Z., Wei, C., Gaidon, A., and Ma, T. Provable guarantees for self-supervised deep learning with spectral contrastive loss. *arXiv preprint arXiv:2106.04156*, 2021.

Hassani, K. and Khasahmadi, A. H. Contrastive multi-view representation learning on graphs. In *International Conference on Machine Learning*, pp. 4116–4126. PMLR, 2020.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

Joshi, C. K., Laurent, T., and Bresson, X. An efficient graph convolutional network technique for the travelling salesman problem. *arXiv preprint arXiv:1906.01227*, 2019.

Karalias, N. and Loukas, A. Erdos goes neural: an unsupervised learning framework for combinatorial optimization on graphs. In *NeurIPS 2020 34th Conference on Neural Information Processing Systems*, 2020.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Kool, W., Van Hoof, H., and Welling, M. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*, 2018.

Kurin, V., Godil, S., Whiteson, S., and Catanzaro, B. Improving SAT Solver Heuristics with Graph Networks and Reinforcement Learning. *CoRR*, abs/1909.11830, 2019.

Lederman, G., Rabe, M. N., Seshia, S., and Lee, E. A. Learning Heuristics for Quantified Boolean Formulas through Reinforcement Learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Nair, V., Bartunov, S., Gimeno, F., von Glehn, I., Lichocki, P., Lobov, I., O'Donoghue, B., Sonnerat, N., Tjandraatmadja, C., Wang, P., et al. Solving mixed integer programs using neural networks. *arXiv preprint arXiv:2012.13349*, 2020.

Nejati, S. and Ganesh, V. Cdcl (crypto) sat solvers for cryptanalysis. In *Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering*, pp. 311–316, 2019.

Nowak, A., Villar, S., Bandeira, A. S., and Bruna, J. Revised note on learning quadratic assignment with graph neural networks. In *2018 IEEE Data Science Workshop (DSW)*, pp. 1–5. IEEE, 2018.

Prates, M., Avelar, P. H., Lemos, H., Lamb, L. C., and Vardi, M. Y. Learning to solve np-complete problems: A graph neural network for decision tsp. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4731–4738, 2019.

Selsam, D. and Bjørner, N. Guiding high-performance sat solvers with unsat-core predictions. In *International Conference on Theory and Applications of Satisfiability Testing*, pp. 336–353. Springer, 2019.

Selsam, D., Lamm, M., Benedikt, B., Liang, P., de Moura, L., Dill, D. L., et al. Learning a sat solver from single-bit supervision. In *International Conference on Learning Representations*, 2018.

Toenshoff, J., Ritzert, M., Wolf, H., and Grohe, M. Graph neural networks for maximum constraint satisfaction. *Frontiers in artificial intelligence*, 3:98, 2021.

Tosh, C., Krishnamurthy, A., and Hsu, D. Contrastive learning, multi-view redundancy, and linear models. In *Algorithmic Learning Theory*, pp. 1179–1206. PMLR, 2021.

Vaezipoor, P., Lederman, G., Wu, Y., Maddison, C., Grosse, R. B., Seshia, S. A., and Bacchus, F. Learning branching heuristics for propositional model counting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):12427–12435, May 2021.

Yolcu, E. and Póczos, B. Learning Local Search Heuristics for Boolean Satisfiability. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pp. 7990–8001, 2019.

You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., and Shen, Y. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33:5812–5823, 2020.

A Machine Learning for Combinatorial Optimization

Supervised Learning for Combinatorial Optimization (CO). Almost all modern approaches to machine learning for CO, use variations of a GNN architecture. On the supervised front, the work of Nowak et al. (2018) on *Quadratic Assignment Problem*, Joshi et al. (2019) and Prates et al. (2019) on *Travelling Salesman Problem* (TSP) have shown encouraging results. Gasse et al. (2019) trained a branching heuristic for MILPs by imitating an expert policy. Later, Nair et al. (2020) extended those ideas to make them scalable to substantially larger instances. In SAT, Selsam et al. (2018) proposed a way to train GNNs to solve SAT problems in an end-to-end fashion. The same architecture was used in Selsam & Bjørner (2019) to guide variable branching across conventional solvers.

Reinforcement Learning for CO. Supervised learning is bottlenecked by the need for labels. Consequently, many have explored the use of *Reinforcement Learning* (RL). Node selection policy of Dai et al. (2017) for TSP and Kool et al. (2018) for *Vehicle Routing Problem* are examples of that effort. Lederman et al. (2020) and Yolcu & Póczos (2019) used REINFORCE to train variable branching heuristic for *quantified Boolean formulas* and local search algorithm WalkSAT, respectively. Kurin et al. (2019) used DQN for SAT and Vaezipoor et al. (2021) improved a SOTA #SAT solver via *Evolution Strategy*. We refer to (Cappart et al., 2021; Bengio et al., 2021) for more comprehensive record of efforts in this area.

Unsupervised Learning for CO. Toenshoff et al. (2021) proposed an unsupervised approach to solve constrained optimization problems on graphs by minimizing a problem dependent loss function. Amizadeh et al. (2019a;b) solved SAT and CircuitSAT problems by minimizing an energy function. Lastly, inspired by probabilistic method, Karalias & Loukas (2020) trained a GNN in an unsupervised way to act as a distribution over possible solutions of a given problem, by minimizing a probabilistic penalty loss. Our method is distinct from other unsupervised techniques in that we do not minimize a problem-dependent loss function and rather learn problem representations through contrastive learning. To the best of our knowledge this is the first attempt at applying contrastive learning in the domain of combinatorial optimization.

B Bipartite Graph Representations of SAT

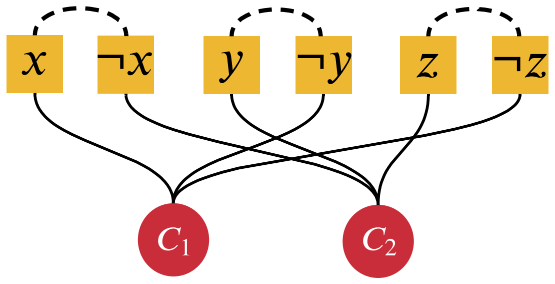

Figure 4: A bipartite graph representation (LIG$^+$) of a SAT formula $\phi := (x \vee \neg y \vee \neg z) \wedge (\neg x \vee y \vee z)$. The subgraph without - - - edges is the LIG of ϕ.

C Descriptions of Each Component in Figure 1

Augmentations. Given a combinatorial instance ϕ, a stochastic augmentation is applied to form a pair of positive samples, denoted by $(\hat{\phi}_1, \hat{\phi}_2)$. Our key insight is the tailored design of augmentations for combinatorial problems should preserve the label of the problem, e.g., satisfiability for SAT.

Format Transformation ($\mathcal{E}$). Formulas $(\hat{\phi}_1, \hat{\phi}_2)$ are transformed into (e.g., LIG$^+$) graphs (G_1, G_2).

Encoder. A neural encoder is trained to extract graph-level representations $\mathbf{z}_1, \mathbf{z}_2 \in \mathbb{R}^d$ from the augmented graphs G_1 and G_2. This encoder can be any GNN such as GCN (Kipf & Welling, 2016) and the encoder proposed by NeuroSAT.

Contrastive Loss. In the end, SimCLR's (Chen et al., 2020a) contrastive loss $\mathcal{L}$ is applied to the graph representations $\{\mathbf{z}_i\}_{i=1}^{2n}$, obtained from a mini-batch of n instances. We follow Chen et al. (2020a) and use an MLP projection head to project each $\mathbf{z}_i$ to $\mathbf{m}_i$. For a positive pair of projected representations $(\mathbf{m}_i, \mathbf{m}_j)$, the loss $\mathcal{L}_{i,j}$ is a cross-entropy loss of differentiating the positive pair from the other $2(n-1)$ negative samples (i.e., augmented samples of other instances):

$$\mathcal{L}_{i,j} = -\log \frac{\exp(sim(\mathbf{m}_i, \mathbf{m}_j) \,/\, \tau)}{\sum_{k=1}^{2N} \mathbf{1}_{k \neq i} \exp(sim(\mathbf{m}_i, \mathbf{m}_k) \,/\, \tau)}, \tag{1}$$

where τ is the temperature parameter, $\mathbf{1}$ is the indicator function, and sim measures the similarity between two representations: $sim(\mathbf{m}_i, \mathbf{m}_j) := \mathbf{m}_i^T \mathbf{m}_j / \|\mathbf{m}_i\| \, \|\mathbf{m}_j\|$. The final loss $\mathcal{L}$ is the average of $\mathcal{L}_{i,j}$ over all positive pairs. After training is completed, we only keep the encoder to extract the representation $\mathbf{z}_i$ for downstream tasks.

D LPAs for SAT

D.1 Label-preserving Augmentations for SAT

The LPAs for SAT preserve satisfiability of any Boolean formula. In other words, a *(un)*satisfiable instance remains *(un)*satisfiable after the applications of LPAs. We review some common LPAs for SAT below, with examples and time complexity results provided in Appendix D.

Unit Propagation (UP). A clause is a *unit clause* if it contains only one literal. If an instance ϕ contains a unit clause ℓ, we can 1) remove all clauses in ϕ containing the literal ℓ and 2) delete $\neg\ell$ from all other clauses.

Add Unit Literal (AU). The inverse of UP: 1) construct a unit clause from a new literal ℓ, 2) add its negation $\neg\ell$ to some other clauses and 3) create new clauses containing ℓ.

Pure Literal Elimination (PL). A variable v is called pure if it occurs with only one polarity in ϕ. We can delete all clauses in ϕ containing v.

Subsumed Clause Elimination (SC). If a clause c_1 is a subset of c_2, i.e., all literals in c_1 are also in c_2, then deleting c_2 does not change satisfiability of ϕ.

Clause Resolution (CR). Resolution produces a new clause implied by two clauses containing complementary literals:

$$\frac{\ell \vee a_1 \vee \cdots \vee a_n, \neg\ell \vee b_1 \vee \cdots b_m}{a_1 \vee \cdots \vee a_n \vee b_1 \vee \cdots b_m} \tag{2}$$

The new clause c is called the *resolvent* of c_1, c_2: $c = c_1 \otimes c_2$. Adding c to ϕ does not change satisfiability.

Variable Elimination (VE). Let S_ℓ be a set of clauses containing the literal ℓ, and $S_{\neg\ell}$ be a set of clauses containing its negation $\neg\ell$. Then a new set S is obtained by pairwise resolving on clauses of S_ℓ and $S_{\neg\ell}$: $S = \{c_1 \otimes c_2 | c_1 \in S_\ell, c_2 \in S_{\neg\ell}\}$. Replacing $S_\ell \cup S_{\neg\ell}$ with S does not change satisfiability Eén & Biere (2005).

Note that some augmentations, such as VE, have the worst-case exponential complexity if run until convergence. However, in our paper, we only eliminate a small and fixed number of variables. Therefore, all LPAs listed here are cheap and have polynomial-time complexity.

D.2 Examples

See Table 1

D.3 Time Complexity

Let n denote the number of clauses in the instance.

Original	UP
$c_1 : x_1$ $c_2 : x_2 \lor x_3$ $c_3 : x_1 \lor \neg x_3 \lor x_4$ $c_4 : \neg x_1 \lor x_2 \lor x_3 \lor \neg x_4$	~~x_1~~ $x_2 \lor x_3$ ~~$x_1 \lor \neg x_3 \lor x_4$~~ ~~$\neg x_1 \lor$~~$x_2 \lor x_3 \lor \neg x_4$
AU	SC
$\neg x_5$ $x_5 \lor x_1$ $x_2 \lor x_3$ $x_1 \lor \neg x_3 \lor x_4$ $\neg x_1 \lor x_2 \lor x_3 \lor \neg x_4$ $\neg x_5 \lor x_1 \lor \neg x_2 \lor x_3$	 x_1 $x_2 \lor x_3$ $x_1 \lor \neg x_3 \lor x_4$ ~~$\neg x_1 \lor x_2 \lor x_3 \lor \neg x_4$~~
CR	VE
x_1 $x_2 \lor x_3$ $x_1 \lor \neg x_3 \lor x_4$ $\neg x_1 \lor x_2 \lor x_3 \lor \neg x_4$ $x_1 \lor x_2 \lor x_4$	x_1 ~~$x_2 \lor x_3$~~ ~~$x_1 \lor \neg x_3 \lor x_4$~~ ~~$\neg x_1 \lor x_2 \lor x_3 \lor \neg x_4$~~ $x_1 \lor x_2 \lor x_4$

Table 1: Examples of LPAs for SAT. UP: remove c_1, c_3 and $\neg x$ from c_4. AU: add unit literal $\neg x_5$, add x_5 to c_1 and create a new random clauses containing $\neg x_5$. SC: remove c_4 because $c_2 \subset c_4$. CR: add $c_2 \otimes c_3$. VE: eliminate x_3 by adding $c_2 \otimes c_3$ and $c_4 \otimes c_3$.

Rule	Explanation	Time Complexity
UP	Eliminate one Unit Clause	$O(n)$
AU	Add one Unit Clause	$O(n)$
PL	Eliminate one pure literal	$O(n)$
SC	Eliminate all subsumed clauses	$O(n^2)$
CR	Resolve one pair of clauses	$O(n)$
VE	Eliminate one variable	$O(n^2)$

Table 2: Complexity results of LPAs

E SETTINGS OF SECTION 4

Architecture. We primarily used the encoder of NeuroSAT (Selsam et al., 2018) as the GNN architecture. Using the NeuroSAT architecture, we obtained the graph-level representations by average-pooling over all literal representations. We followed the design details of the original NeuroSAT paper.

Experimental Setting. We used the contrastive loss in Equation 1 with the temperature 0.5. We used Adam optimizer with learning rate 2×10^{-4} and weight decay 10^{-5}. The batch size was 128 and the maximum training epoch was 5000. The generator produced a set of new unlabelled instances for each batch. We used 100 labelled instances to train our linear evaluators, and another 500 as the validation set to pick the hyperparameters (ranging from 10^{-3} to 10^{3}) of L_2 regularization. The test set consisted of 10^4 instances.

Datasets. We experimented using four generators: SR (Selsam et al., 2018), Power Random 3SAT (PR) (Ansótegui et al., 2009), Double Power (DP) and Popularity Similarity (PS) (Giráldez-Cru & Levy, 2017). SR and PR are the synthetic generators. DP and PS are pseudo-industrial generators producing instances that mimic real-world problems. We generated instances of 10 variables for SR and PR, and 20 for DP and PS. The number inside the parenthesis denotes the number of variables per instance. We also tweaked the parameters of the generators so that they produce roughly balanced SAT and UNSAT. More details are in Appendix H.

F The heatmap for LAAs

See 5.

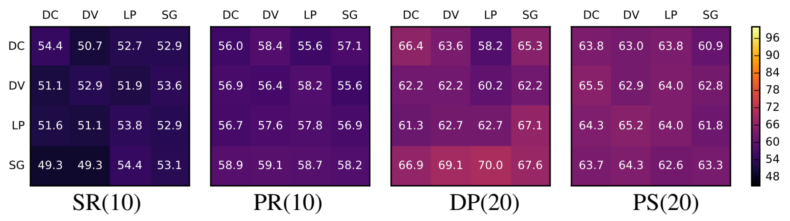

Figure 5: Heatmap: linear evaluation accuracy of SSL models trained with different single LAA augmentations (diagonal) or paired combinations (off-diagonal). Each column denotes different datasets. For off-diagonal entries on each heatmap, the row corresponds to the first augmentation applied.

G Settings of Section 5

The details of training SSL models follow Section 4. For supervised models, we used the same architecture as NeuroSAT. The hyperparameters for supervised were the same as SSL in Section 4, except the learning rate was chosen to be 2×10^{-5}, following Selsam et al. (2018). For each dataset, we chose the best augmentation combination for LPA and LAA according to Figure 2. The degree parameter associated with each augmentation was tuned separately for SSL and supervised models. We used 200 instances as validation sets for early stopping of all methods.

Unless otherwise specified, all datasets used in this section have 40 variables per instance. For SR datasets, following NeuroSAT, we trained on SR(U(10, 40)) and tested on SR(40). Training procedures for all models were the same as Section E, except the batch size was 80 for SR(40) to avoid memory issues. When fine-tuning NeuroSAT, we used different learning rates for the encoder and aggregator, which were separately tuned for different models on each dataset.

H Datasets

H.1 Brief Description

- **SR (Selsam et al., 2018)** A random SAT generator proposed as a challenge for neural networks to learn intrinsic properties about satisfiability without cheating on some miscellaneous statistics about the dataset.
- **PR (Ansótegui et al., 2009)** A random k-SAT generator where the frequency of each variable is sampled from a power-law distribution.
- **DP (Ansótegui et al., 2009)** A pseudo-industrial generator based on PR with varying clause length
- **PS (Giráldez-Cru & Levy, 2017)** A pseudo-industrial generator based on the notion of locality.

H.2 Parameters

For the purpose of reproducibility, we show the parameters of each generator used in the paper below:

- **SR(10) / SR(40)** All parameters follow NeuroSAT.
- **PR(10)** Number of variables: 10. Number of clauses: 41. Variable per clause: 3. Power-law exponents of variables: 1.7.
- **PR(40)** Number of variables: 40. Number of clauses: 147. Variable per clause: 3. Power-law exponents of variables: 2.5.

- **DP(20)** Number of variables: 20. Number of clauses: 34. Average variables per clause: 4. Power-law exponents of variables: 1.7.
- **DP(40)** Number of variables: 40. Number of clauses: 75. Average variables per clause: 5. Power-law exponents of variables: 1.7.
- **PS(20)** Number of variables: 20. Number of clauses: 58. Min variable per clause: 2. Average variables per clause: 4.
- **PS(40)** Number of variables: 40. Number of clauses: 73. Min variable per clause: 2. Average variables per clause: 5.