Learning Better Representations From Less Data For Propositional Satisfiability

Mohamed Ghanem* **Frederik Schmitt*** **Julian Siber*** **Bernd Finkbeiner***
*CISPA Helmholtz Center for Information Security
{mohamed.ghanem,frederik.schmitt,julian.siber,finkbeiner}@cispa.de

Abstract

Training neural networks on NP-complete problems typically demands very large amounts of training data and often needs to be coupled with computationally expensive symbolic verifiers to ensure output correctness. In this paper, we present NeuRes, a neuro-symbolic approach to address both challenges for propositional satisfiability, being the quintessential NP-complete problem. By combining certificate-driven training and expert iteration, our model learns better representations than models trained for classification only, with a much higher data efficiency – requiring orders of magnitude less training data. NeuRes employs propositional resolution as a proof system to generate proofs of unsatisfiability and to accelerate the process of finding satisfying truth assignments, exploring both possibilities in parallel. To realize this, we propose an attention-based architecture that autoregressively selects pairs of clauses from a dynamic formula embedding to derive new clauses. Furthermore, we employ expert iteration whereby model-generated proofs progressively replace longer teacher proofs as the new ground truth. This enables our model to reduce a dataset of proofs generated by an advanced solver by $\sim 32\%$ after training on it with no extra guidance. This shows that NeuRes is not limited by the optimality of the teacher algorithm owing to its self-improving workflow. We show that our model achieves far better performance than NeuroSAT in terms of both correctly classified and proven instances.

1 Introduction

Boolean satisfiability (SAT) is a fundamental problem in computer science. For theory, this stems from SAT being the first problem proven NP-complete [13]. For practice, this is due to many highly-optimized SAT solvers being used as flexible reasoning engines in a variety of tasks such as model checking [12, 47], software verification [17], planning [27], and mathematical proof search [24]. Recently, SAT has also served as a litmus test for assessing the symbolic reasoning capabilities of neural models and a promising domain for neuro-symbolic systems [43, 42, 1, 10, 36]. So far, neural models only provide limited, if any, justification for unsatisfiability predictions. NeuroCore [42], for example, predicts an unsatisfiable core, the verification of which can be as hard as solving the original problem. No certificates at all or certificates that are hard to check limit neural methods in a domain where correctness is critical and prevents close integrations with symbolic methods. Therefore, we propose a neuro-symbolic model that utilizes *resolution* to solve SAT problems by generating easy-to-check certificates.

A resolution proof is a sequence of case distinctions, each involving two clauses, that ends in the empty clause (falsum). This technique can also be used to prove satisfiability by exhaustively applying it until no further new resolution steps are possible and the empty clause has not been derived. Generating such a proof is an interesting problem from a neuro-symbolic perspective because unlike other discrete combinatorial problems that have been considered before [46, 7, 28, 30, 11], it

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

 https://doi.org/10.52202/079017-1532

requires selecting compatible *pairs* of clauses from the dynamically growing pool, as newly derived clauses are naturally considered for derivation in subsequent steps. In this work, we devise three attention-based mechanisms to perform this pair-selection needed for generating resolution proofs. In addition, we augment the architecture to efficiently handle *sat* (satisfiable) formulas with an assignment decoding mechanism that assigns a truth value to each literal. We hypothesize that, despite their final goals being in complete opposition, resolution and *sat* assignment finding can form a mutually beneficial collaboration. On the one hand, clauses derived by resolution incrementally inject additional information into the network, e.g., deriving a single-literal clause by resolution directly implies that literal should be true in any possible *sat* assignment. On the other hand, finding a *sat* assignment absolves the resolution network from having to prove satisfiability by exhaustion. On that basis, given an input formula, NeuRes proceeds in two parallel tracks: (1) finding a *sat* assignment, and (2) deriving a resolution proof of unsatisfiability. Both tracks operate on a shared representation of the problem state. Depending on which track succeeds, NeuRes produces the corresponding SAT verdict which is guaranteed to be sound by virtue of its certificate-based design. Since both of our certificate types are efficient to check, we can afford to perform these symbolic checks at each step. When comparing NeuRes with NeuroSAT [43], which has been trained to predict satisfiability with millions of samples, we demonstrate that NeuRes achieves a higher accuracy while providing a proof and requires only thousands of training samples.

As for most problems in theorem proving we are not only interest in finding any proof but a short proof. Resolution proofs can vary largely in their size depending on the resolution steps taken. Being able to efficiently check the proof, also allows us to adapt the proof target while training the model. In particular, we explore an expert iteration mechanism [2, 39] that pre-rolls the resolution proof of the model and replaces the target proof whenever the pre-rolled proof is shorter. We demonstrate that this bootstrapping mechanism iteratively shortens the proofs of our training dataset while further improving the overall performance of the model.

We make the following contributions:

1. We introduce novel architectures which combine graph neural networks with attention mechanisms for generating resolution proofs and assignments for CNF formulas (Section 4).
2. We show that for propositional logic, learning to prove rather than predict satisfiability results in better representations and requires far less training samples (Section 6 and 7).
3. We devise a bootstrapped training procedure where our model progressively produces shorter resolution proofs than its teacher (Section 6.2) boosting the model's overall performance.

The implementation of our framework can be found at `https://github.com/Oschart/NeuRes`.

2 Related Work

SAT Solving and Certificates. We refer to the annual SAT competitions [5] for a comprehensive overview on the ever-evolving landscape of SAT solvers, benchmarks, and proof checkers. SAT solvers are complex systems with a documented history of bugs [9, 26], hence proof certificates have been partially required in this competition since 2013 [3]. Unlike satisfiable formulas, there are several ways to certify unsatisfiable formulas [23]. Resolution proofs [52, 20] are easy to verify [15], but non-trivial to generate from modern solvers based on the paradigm of conflict-driven clause learning [34]. Clausal proofs, e.g., in DRAT format [50], are easier to generate and space-efficient, but hard to validate. Verifying the proofs can take longer than their discovery [22] and requires highly optimized algorithms [31].

Deep Learning for SAT Solving. NeuroSAT [43] was the first study of the Boolean satisfiability problem as an end-to-end learning task. Building upon the NeuroSAT architecture, a simplified version has been trained to predict unsatisfiable cores and successfully integrated as a branching heuristic in a state-of-the-art SAT solver [42]. Recent work has employed a related architecture as a phase selection heuristic [49]. It has been shown that both the NeuroSAT architecture and a newly introduced deep exchangeable architecture can outperform SAT solvers on instances of 3-SAT problems [10]. The NeuroSAT architecture has also been applied on special classes of crypto-analysis problems [44]. In addition to supervised learning, unsupervised methods have been proposed for solving SAT problems. For Circuit-SAT a deep-gated DAG recursive neural architecture has been

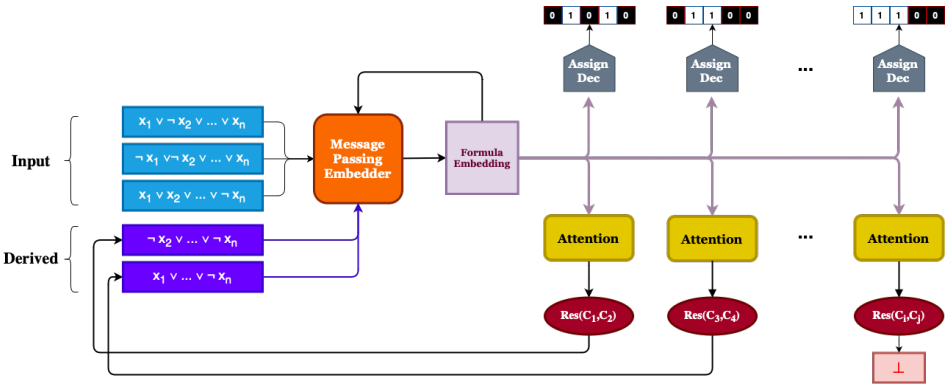


Figure 1: Overall NeuRes architecture

presented together with a differentiable training objective to optimize towards solving the Circuit-SAT problem and finding a satisfying assignment [1]. For Boolean satisfiability, a differentiable training objective has been proposed together with a query mechanism that allows for recurrent solution trials [36].

Deep Learning for Formal Proof Generation. In formal mathematics, deep learning has been integrated with theorem proving for clause selection [33, 18], premise selection [25, 48, 6, 35], tactic prediction [51, 37] and whole proof searches [40, 19]. For SMT formulas specifically, deep reinforcement learning has been applied to tactic prediction [4]. In the domain of quantified boolean formulas, heuristics have been learned to guide search algorithms in proving the satisfiability and unsatisfiability of formulas [32]. For temporal logics, deep learning has been applied to prove the satisfiability of linear-time temporal logic formulas and the realizability of specifications [21, 41, 14].

3 Proofs of (Un-)Satisfiability

We start with a brief review of certifying the (un-)satisfiability of propositional formulas in conjunctive normal form. For a set of Boolean variables V, we identify with each variable $x \in V$ the *positive literal* x and the *negative literal* $\neg x$ denoted by $\bar{x}$. A *clause* corresponds to a disjunction of literals and is abbreviated by a set of literals, e.g., $\{\bar{1}, 3\}$ represents $(\neg x_1 \vee x_3)$. A formula in *conjunctive normal form (CNF)* is a conjunction of clauses and is abbreviated by a set of clauses, e.g., $\{\{\bar{1}, 3\}, \{1, 2, \bar{4}\}\}$ represents $(\neg x_1 \vee x_3) \wedge (x_1 \vee x_2 \vee \neg x_4)$. Any Boolean formula can be converted to an equisatisfiable CNF formula in polynomial time, for example with Tseitin transformation [45].

A CNF formula is *satisfiable* if there exists an *assignment* $\mathcal{A} : V \to \{\top, \bot\}$ such that all clauses are satisfied, i.e., each clause contains a positive literal x such that $\mathcal{A}(x) = \top$ or a negative literal $\bar{x}$ such that $\mathcal{A}(x) = \bot$. If no such assignment exists we call the formula *unsatisfiable*. To prove unsatisfiability we rely on resolution, a fundamental inference rule in satisfiability testing [16]. The resolution rule (*Res*) picks clauses with two opposite literals and performs the following inference:

$$\frac{C_1 \cup \{x\} \qquad C_2 \cup \{\bar{x}\}}{C_1 \cup C_2}\ \textit{Res}$$

Resolution effectively performs a case distinction on the value of variable x: Either it is assigned to false, then C_1 has to evaluate to true, or it is assigned to true, then C_2 has to evaluate to true. Hence, we may infer the clause $C_1 \cup C_2$. A *resolution proof* for a CNF formula is a sequence of applications of the *Res* rule ending in the empty clause.

4 Models

4.1 General Architecture

NeuRes is a neural network that takes a CNF formula as a set of clauses and outputs either a satisfying truth assignment or a resolution proof of unsatisfiability. As such, our model comprises a formula

https://doi.org/10.52202/079017-1532

embedder connected to two downstream heads: (1) an attention network responsible for selecting clause pairs, and (2) a truth assignment decoder. See Figure 1 for an overview of the NeuRes architecture. After obtaining the initial clause and literal embeddings (representing the input formula), we continue with the iterative certificate generation phase. At each step, the model selects a clause pair which gets resolved into a new clause to append to the current formula graph while decoding a candidate truth assignment in parallel. The model keeps deriving new clauses until the empty clause is found (marking resolution proof completion), a satisfying assignment is found (marking a certified *sat* verdict), or the limit on episode length is reached (marking timeout).

4.2 Message-Passing Embedder

Similar to NeuroSAT, we use a message-passing GNN to obtain clause and literal embeddings by performing a predetermined number of rounds. Our formula graph is also constructed in a similar fashion to NeuroSAT graphs where clause nodes are connected to their constituent literal nodes and literals are connect to their complements (cf. Appendix A). For a formula in m variables and n clauses, the outputs of this GNN are two matrices: $E^L \in \mathbb{R}^{m \times d}$ for literal embeddings and $E^C \in \mathbb{R}^{n \times d}$ for clause embedding, where $d \in \mathbb{N}^+$ is the embedding dimension. Here we have two key differences from NeuroSAT. Firstly, NeuroSAT uses these embeddings as voters to predict satisfiability through a classification MLP. In our case, we use these embeddings as clause tokens for clause pair selection and literal tokens for truth value assignment. Secondly, since our model derives new clauses with every resolution step, we need to embed these new clauses, as well as update existing embeddings to reflect their relation to the newly inferred clauses. Consequently, we need to introduce a new phase to the message-passing protocol, for which we explore two approaches: *static embeddings* and *dynamic embeddings*.

In a *static* approach, we do not change the embeddings of initial clauses upon inferring a new clause. Instead, we exchange local messages between the node corresponding to the new clause and its literal nodes, in both directions. The main advantage of this approach is its low cost. A major drawback is that initial clauses never learn information about their relation to newly inferred clauses.

In a *dynamic* approach, we do not only generate a new clause and its embedding, we also update the embeddings of all other clauses. This accounts for the fact that the utility of an existing clause may change with the introduction of a new clause. We perform one message-passing round on the mature graph for every newly derived clause, which produces the new clause embedding and updates other clause embeddings. Since message-passing rounds are parallel across clauses, a single update to the whole embedding matrix is reasonably efficient.

4.3 Selector Networks

After producing clause and literal embeddings, NeuRes enters the derivation stage. At each step, our model needs to select two clauses to resolve, produce the resultant clause, and add it to the current formula. To realize our clause-pair selection mechanism, we employ three attention-based designs.

4.3.1 Cascaded Attention (Casc-Attn)

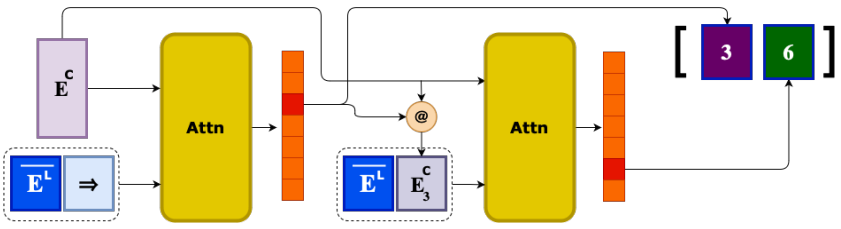


Figure 2: Cascaded attention

In this design, pairs are selected by making two consecutive attention queries on the clause pool. We condition the second attention query on the outcome (i.e., the clause) of the first query. Figure 2 shows this scheme where we perform the first query using the mean of the literal embeddings $\overline{E^L}$ concatenated with a zero vector while performing the second query using the mean of the literal

embeddings concatenated with the embedding vector $E^C_{c_1}$ of the clause selected in the first query. Formally, Casc-Attn selects a clause index pair (c_1, c_2) as follows:

$$c_i = \underset{j}{\textbf{argmax}} \left[u^T \tanh(W_1 q_i + W_2 E^C_j) \right] \quad \text{with} \quad q_i = \begin{cases} \overline{E^L} \parallel \mathbf{0} & \text{if } i = 1 \\ \overline{E^L} \parallel E^C_{c_1} & \text{if } i = 2 \end{cases} \tag{1}$$

where $W_1 \in \mathbb{R}^{2d \times d}, W_2 \in \mathbb{R}^{d \times d}, u \in \mathbb{R}^d$ are trainable network parameters.

The advantage of this design is that it is not limited to pair selection and can be used to select a tuple of arbitrary length. The main downside, however, is that this design chooses c_1 independently from c_2, which is undesirable because the utility of a resolution step is determined by both clauses simultaneously (not sequentially).

4.3.2 Full Self-Attention (Full-Attn)

To address the downside of independent clause selection, this variant performs self-attention between all clauses to obtain a matrix $S \in \mathbb{R}^{n \times n}$ where $S_{i,j}$ represents the attention score of the clause pair (c_i, c_j) as shown in Figure 3. The model selects clause pairs by choosing the cell with the maximal score. In this attention scheme, the clause embeddings are used as both queries and keys.

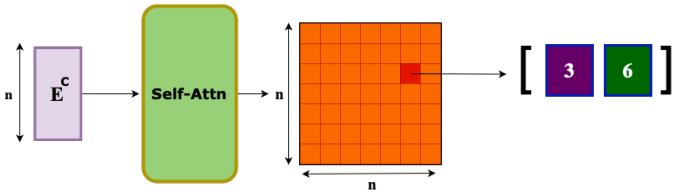


Figure 3: Full self-attention

Formally, Full-Attn selects a clause index pair (c_1, c_2) as follows:

$$(c_1, c_2) = \underset{(i,j)}{\textbf{argmax}} \; S_{i,j} \quad \text{with} \quad Q = E^C W_Q; \quad K = E^C W_K; \quad S = \frac{QK^T}{\sqrt{d}} \tag{2}$$

where $W_Q \in \mathbb{R}^{d \times d}, W_K \in \mathbb{R}^{d \times d}$ are trainable network parameters. Since S contains many cells that correspond to invalid resolution steps (i.e., clause pairs that cannot be resolved), we mask out the invalid cells from the attention grid in ensure the network selection is valid at every step.

4.3.3 Anchored Self-Attention (Anch-Attn)

In Full-Attn, the attention grid grows quadratically with the number of clauses. In this variant, we relax this cost by exploiting a property of binary resolution where each step targets a single variable in the two resolvent clauses. This allows us to narrow down candidate clause pairs by first selecting a variable as an anchor on which our clauses should be resolved. As such, we do not need to consider the full clause set at once, only the clauses containing the chosen variable v. We further compress the attention grid by lining clauses containing the literal v on rows while lining clauses containing the literal $\neg v$ on columns. This reduces the redundancy of the attention grid since clauses containing the variable v with the same parity cannot be resolved on v, so there is no point in matching them. In this scheme, we have two attention modules: one attention network to choose an anchor variable followed by a self-attention network to produce the anchored score grid.

In light of Figure 4, this approach combines structural elements from Casc-Attn and Full-Attn; however, both elements are used differently in Anch-Attn. Firstly, the attention mechanism in Casc-Attn is used to select clauses whereas Anch-Attn uses it to select variables. Secondly, self-attention in Full-Attn matches any pair of clauses (c_i, c_j) in both directions as the row and column dimensions in the attention score grid reflect the same clauses (all clauses). By contrast, Anch-Attn computes self-attention scores for clause pairs in only one order (positive instance to negative instance). Formally,

https://doi.org/10.52202/079017-1532

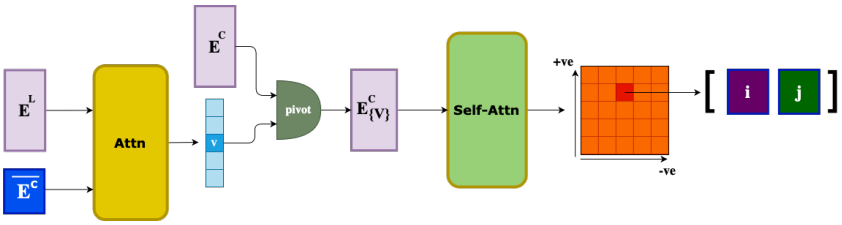


Figure 4: Anchored self-attention

Anch-Attn selects an anchor variable v as follows:

$$v = \underset{i}{\textbf{argmax}} \left[u^T \tanh(W_1 \overline{E^C} + W_2(E_i^{L^+} + E_i^{L^-})) \right] \tag{3}$$

where $W_1 \in \mathbb{R}^{d \times d}, W_2 \in \mathbb{R}^{d \times d}, u \in \mathbb{R}^d$ are trainable network parameters. The clause index pair (c_1, c_2) is then selected according to the same equations of Full-Attn (Eq. 2) using the v-anchored set of clause embeddings.

4.4 Assignment Decoder

To extract satisfying assignments, we use a sigmoid-activated MLP ψ on top of the literal embeddings E^L to assign a truth value $\hat{\mathcal{A}}(l_i)$ to a literal l_i as shown in Eq. 4.

$$\hat{\mathcal{A}}(l_i) = \sigma(\psi(E_i^L)) \tag{4}$$

Note that since for each variable, we have a positive and a negative literal embeddings, we can construct two different truth assignments at a time using this method. However, supervising both assignments did not improve the performance compared to only supervising the positive assignment (on positive literal embeddings). Thus, to simplify our loss function, we only derive truth assignments from the positive literal embeddings at train time while extracting both at test time. Interestingly, at test time, we found using negative literals (in addition to the positive ones) sometimes produces satisfying assignments before the positive branch despite receiving no direct supervision during training. Our intuition regarding this observation attributes it to the fact that the formula graph has no explicit notion of positive and negative literals, it only represents connections to clauses (positive and negative literals are connected by an undirected edge that does not distinguish their parity). As such, both literal nodes have a different local view into the rest of formula, which could result in one of them leading to a satisfying assignment faster than the other.

5 Training and Hyperparameters

5.1 Dataset

For our training and testing data, we adopt the same formula generation method as NeuroSAT [43], namely **SR**(n) where n is the number of variables in the formula. This method was designed to generate a generalized formula distribution that is not limited to a particular domain of SAT problems. To control our data distributions, we vary the range on the number of Boolean variables involved in each formula. For our training data, we use formulas in **SR**$(U(10, 40))$ where $U(10, 40)$ denotes the uniform distribution on integers between 10 and 40 (inclusive). To generate our teacher certificates comprising resolution proofs and truth assignments, we use the BooleForce solver [8] on the formulas generated on the **SR** distribution.

5.2 Loss Function

We train our model in a supervised fashion using teacher-forcing on solver certificates. During *unsat* episodes, teacher actions (clause pairs) are imposed over the whole run. The length of the teacher proof dictates the length of the respective episode, denoted as T. Model parameters θ maximize the likelihood of teacher choices y_t thereby minimizing the resolution loss $\mathcal{L}_{Res}$ shown in Eq 5.

$$\mathcal{L}_{Res} = -\frac{1}{T} \sum_t \log(p(y_t; \theta)) \cdot \gamma^{(T-t)} \tag{5}$$

During *sat* episodes, we minimize $\mathcal{L}_{sat}$ computed as the binary cross-entropy loss between the sigmoid-activated outputs of assignment decoder $\hat{\mathcal{A}} : V \to [0,1]$ and the teacher assignment $\mathcal{A} : V \to \{0,1\}$ as shown in Eq. 6.

$$\mathcal{L}_{sat} = \frac{1}{T}\sum_{t}\left[\frac{\gamma^{(T-t)}}{|V|}\sum_{v}^{V}\text{BCE}(\hat{\mathcal{A}}(v), \mathcal{A}(v))\right] \tag{6}$$

In both types of episodes, step-wise losses are weighted by a time-horizon discounting factor $\gamma < 1.0$ over the whole episode. The main rationale behind this is that later losses should have higher weights as the formula tends to get easier to solve with each new clause inferred by resolution.

5.3 Hyperparameters

NeuRes has several hyperparameters that influence network size, depth, and loss weighting. In the experiments we fix the embedding dimension to 128. We train our models with a batch size of 1 and the Adam optimizer [29] for 50 epochs which took about six days on a single NVIDIA A100 GPU. We linearly anneal the learning rate from 5×10^{-5} to zero over the training episodes. This empirically yields better results than using a constant learning rate. We use a time discounting factor $\gamma = 0.99$ for the episodic loss. We apply global-norm gradient clipping with a ratio of 0.5 [38].

6 Generating Resolution Proofs

NeuRes uses resolution as the core reasoning technique for certificate generation, both in the *unsat* and *sat* cases. Hence, we start with an in-depth comparative evaluation of several internal variants for resolution only. In particular, we evaluate the success rate (i.e., problems solved before timeout) and proof length relative to the teacher, denoted by p-Len $= \frac{|\mathcal{P}_{\text{NeuRes}}|}{|\mathcal{P}_{\text{teacher}}|}$. We use a limit of 4 on this ratio as a timeout to avoid simply brute-forcing a resolution proof. Note that we measure p-Len only for solved formulas to avoid diluting the average with resolution trails that timed out. For experiments in this section, we train on 8K *unsat* formulas in $\mathbf{SR}(U(10,40))$ and test our models on 10K unseen formulas belonging to the same distribution. We use more formulas than the model was trained on to more reliably demonstrate its learning capacity.

Table 1: Performance of all attention variants on *unsat* $\mathbf{SR}(U(10,40))$ test problems.

VARIANT	STATIC-EMBED		DYNAMIC-EMBED	
	PROVEN (%)	P-LEN	PROVEN (%)	P-LEN
CASC-ATTN	14.72	1.87	37.33	1.79
FULL-ATTN	25.38	**1.61**	**95.2**	**1.67**
ANCH-ATTN	**28.72**	2.12	60.5	2.28

6.1 Attention Variants

To assess the basic resolution performance of NeuRes, we evaluate each attention variant using both static and dynamic embeddings. For this experiment, we perform 32 rounds of message-passing for each input formula. As shown in Table 1, dynamic embedding is decisively better for all three attention variants, thereby confirming its conceptual merit. While anchored attention leads over other variants under static embeddings, full attention performs significantly better for dynamic embeddings, albeit at the cost of longer proofs on average. We believe that Anch-Attn's better performance in the static setting can be explained through the full connectivity of its attention grid (proven in Appendix B). Since dynamic-embedding Full-Attn is the best-performing configuration over in-distribution test settings, we will demonstrate the remaining evaluation experiments exclusively on this variant.

https://doi.org/10.52202/079017-1532

Table 2: Bootstrapped training data reduction statistics. Reduction statistics are computed on the $\mathbf{SR}(U(10, 40))$ training set while p-Len and success rate are computed on a test set of the same distribution.

REDUCTION DEPTH	**MAX**: 23, **AVG**: 6.6
PROOF REDUCTION (%)	**MAX**: 86.11, **AVG**: 33.51
PROOFS REDUCED (%)	90.08
TOTAL REDUCTION (%)	31.85
P-LEN	1.15
SUCCESS RATE (%)	100.0

6.2 Shortening Teacher Proofs with Bootstrapping

During our initial experiments, we discovered proofs produced by NeuRes that were shorter than the corresponding teacher proofs in the training data. Although teacher proofs were generated by a traditional SAT solver, they are not guaranteed to be size-optimal. The size of resolution proofs is their only real drawback, hence any method that can reduce this size would be immensely useful. Upon closer inspection we find that, on average, our previous best performer trained with regular teacher-forcing manages to shorten $\sim 18\%$ of teacher proofs by a notable factor (cf. Appendix C). This inspired us to devise a bootstrapped training procedure to capitalize on this feature: We pre-roll each input problem using model actions only, and whenever the model proof is shorter than the teacher's, it replaces the teacher's proof in the dataset. In other words, we maximize the likelihood of the shorter proof. In doing so iteratively, the model progressively becomes its own teacher by exploiting redundancies in the teacher algorithm.

The outcome of this bootstrapped training process is summarized in Table 6. We find that bootstrapping results in notable gains in terms of both success rate and optimality. The sharp decline in proof length (relatively quantified by p-Len) at test time shows that the models transfers the bootstrapped knowledge to unseen test formulas, as opposed to merely overfitting on training formulas. In addition to success rate and p-Len, we inspect the reduction statistics of our bootstrapped variant (first three rows of Table 6). Since the bootstrapped model performs multiple reduction scans over the training dataset, we add a metric for reduction depth computed as the number of progressive reductions made to a proof. To further quantify this effect, we report the maximum and average reduction ratios of reduced proofs relative to teacher proofs. Finally, we report the total reduction made to the dataset size in terms of total number of proof steps.

In Appendix C, we have compiled additional statistics (cf. Table 5) on proof shortening during the training process, as well as an example proof reduced by the bootstrapped NeuRes (Figure 7). We only include a small reduction example (from 20 steps to 10 steps) for space constraints, but we observed many more examples of much larger reductions (e.g., over 400 steps).

7 Resolution-Aided SAT Solving

In this section, we evaluate the performance of our fully integrated model trained on a hybrid dataset comprising 8K unsatisfiable formulas (and their resolution proofs) and 8K satisfiable formulas (and their satisfying assignments). For the unsatisfiable formulas, timeout ($4 \times |\mathcal{P}_{\text{teacher}}|$) and optimality (p-Len) are measured similarly to previous experiments. For satisfiable formulas, we set the timeout (maximum #trials) to $2 \times |V|$. Ultimately, this section aims to investigate the effect of incorporating a certificate-driven downstream head on the quality of the learnt representations through its impact on the performance of the complementary task, i.e., proving/predicting satisfiability. We use NeuroSAT as our baseline as it employs the same formula embedding architecture. Since NeuroSAT proves *sat* but only *predicts unsat*, we train a classification MLP on top of our trained NeuRes model to further showcase the benefit of our representations on prediction accuracy.

Table 3 confirms this main hypothesis. In essence, this result points to the fact that learning signals obtained from training on *unsat* certificates largely enhance the ability of the neural network to extract useful information from the input formula. This is doubly promising considering NeuroSAT was

Table 3: Performance of full solver mode tested on $\mathbf{SR}(40)$ problems and trained on $\mathbf{SR}(U(10, 40))$ problems where PREDICTED refers to the satisfiability prediction without certificate.

MODEL	PROVEN (%)			PREDICTED (%)		
	SAT	UNSAT	TOTAL	SAT	UNSAT	TOTAL
NEURES	**96.8**	**99.6**	**98.2**	**84.28**	**99.2**	**91.65**
NEUROSAT [43]	70	-	-	73	96	85

trained on *millions* of formulas while NeuRes was trained on only *16K* formulas. Lastly, we find that augmenting *sat* formulas by resolution derivations results in relative improvements ($\sim 2.3\%$) in success rate even though these derivations are attempting to prove unsatisfiability.

8 Utilizing Model Fan-Out

In our Full-Attention module, we compute n^2 scores and only perform the top-score resolution step. This greedy approach arguably underutilizes the attention grid computations as it ignores other high-scoring steps that might lead to a shorter proof thereby improving the success rate in addition to reducing the number of queries to the model. The latter leads to an overall runtime reduction since performing an extra symbolic resolution step is much faster than a forward model pass. As such, we experiment with performing the top k steps of the attention grid after each forward pass. It should be noted, however, that this yields diminishing returns as it leads to a faster growth of the clause base which in turn inflates the attention grid. For $k > 1$, after deriving the empty clause, only clauses that connect to it in the resolution graph are kept in the final proof. This post-processing step is linear in the proof length and eliminates redundant resolution steps resulting from the higher fan-out. Table 4 shows that taking the top 3 steps already yields a massive reduction in proof lengths along with a significant boost to the success rate.

Table 4: Performance of different model fan-outs on $\mathbf{SR}(40)$ test data. Proof length (p-Len) and #Model Calls are both normalized by the length of the teacher proof.

FULL-ATTN FAN-OUT	P-LEN	#MODEL CALLS	TOTAL PROVEN (%)
TOP-1	1.15	1.15	98.2
TOP-3	0.57	0.49	99.9
TOP-5	**0.52**	**0.43**	**100.0**

One way to offset the attention grid inflation with higher fan-out would be to keep a saliency map for all clauses then discarding k clauses with the least saliency scores after each forward pass. One simple way to compute this saliency score for a clause would be the sum/mean/max of its respective row in the attention scores grid. Another proxy for saliency could be the recency score reflecting how many steps have elapsed since the last time a given clause was used.

9 Generalizing to Larger Problems

In order to test our model's out-of-distribution performance, we evaluate our NeuRes model on five datasets comprising formulas with up to 5 times more variables than encountered during training. We use the same distributions reported by NeuroSAT and we run our model for the same maximum number of iterations (1000).

Figure 5 shows the scalability of NeuRes to larger problems by letting it run for more iterations. Compared to NeuroSAT [43], NeuRes scores a much higher first-try success rate on all 5 problem distributions, and a higher final success rate on all of them except for $\mathbf{SR}(40)$ on which both models nearly score 100%. Particularly, NeuRes shows higher first-try success on the 3 largest problem sizes where NeuroSAT solves zero or near-zero problems on the first try.

https://doi.org/10.52202/079017-1532

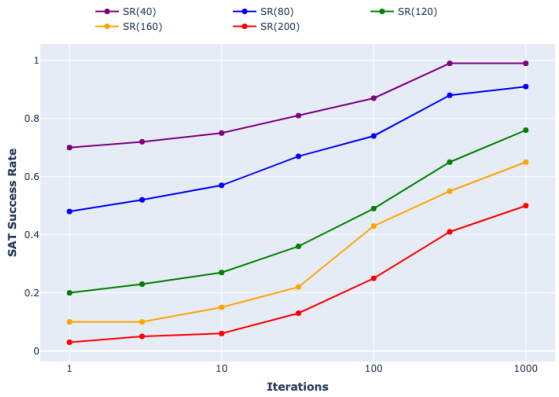


Figure 5: SAT success rate over iterations.

10 Conclusion

In this paper, we introduced a deep learning approach for proving and predicting propositional satisfiability. We proposed an architecture that combines graph neural networks with attention mechanisms to generate resolution proofs of unsatisfiability. Unlike methods that merely predict unsatisfiability, our models provide easily verifiable certificates for their verdicts. We demonstrated that our certificate-based training and resolution-aided mode of operation surpass previous approaches in terms of performance and data efficiency, which we attribute to learning better representations.

Despite its promising benchmark performance, our model cannot solely outperform highly engineered industrial solvers, as is currently the case for all neural methods as standalone tools. The gap between neural networks and symbolic algorithms is still rather large, and our hope is to bring deep learning methods one concrete step closer to filling this gap. For NeuRes, this step is recognizing the immense value of carefully integrating certificates into the model design and training as opposed to using shallow supervision labels. Last but not least, it is worth noting that even at their present state, neural networks stand great potential to advance traditional solvers by combining them into hybrid solvers that utilize the deep long-range dependencies captured by neural networks along with the exploration speed of symbolic algorithms. Moreover, we demonstrated a unique potential to advance SAT solving through proof reduction, as proof size is a major challenge in certifying the results of traditional solvers. This proof reduction is facilitated by a bootstrapped training procedure that uses teacher proofs as a guide as opposed to a golden standard.

Acknowledgments and Disclosure of Funding

This work was supported by the European Research Council (ERC) Grant HYPER (No. 101055412).

References

[1] S. Amizadeh, S. Matusevych, and M. Weimer. Learning to solve circuit-sat: An unsupervised differentiable approach. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[2] T. Anthony, Z. Tian, and D. Barber. Thinking fast and slow with deep learning and tree search. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5360–5370, 2017.

[3] A. Balint, A. Belov, M. Heule, and M. Järvisalo, editors. *Proceedings of SAT Competition 2013: Solver and Benchmark Descriptions*, volume B-2013-1 of *Department of Computer Science Series of Publications B*. University of Helsinki, Finland, 2013.

[4] M. Balunovic, P. Bielik, and M. T. Vechev. Learning to solve SMT formulas. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 10338–10349, 2018.

[5] T. Balyo, M. Heule, M. Iser, M. Järvisalo, and M. Suda, editors. *Proceedings of SAT Competition 2023: Solver, Benchmark and Proof Checker Descriptions*. Department of Computer Science Series of Publications B. Department of Computer Science, University of Helsinki, Finland, 2023.

[6] K. Bansal, S. M. Loos, M. N. Rabe, C. Szegedy, and S. Wilcox. Holist: An environment for machine learning of higher order logic theorem proving. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 454–463. PMLR, 2019.

[7] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio. Neural combinatorial optimization with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017.

[8] A. Biere. Booleforce sat solver. `https://fmv.jku.at/booleforce/`, 2010.

[9] R. Brummayer, F. Lonsing, and A. Biere. Automated testing and debugging of SAT and QBF solvers. In O. Strichman and S. Szeider, editors, *Theory and Applications of Satisfiability Testing - SAT 2010, 13th International Conference, SAT 2010, Edinburgh, UK, July 11-14, 2010. Proceedings*, volume 6175 of *Lecture Notes in Computer Science*, pages 44–57. Springer, 2010.

[10] C. Cameron, R. Chen, J. S. Hartford, and K. Leyton-Brown. Predicting propositional satisfiability via end-to-end learning. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 3324–3331. AAAI Press, 2020.

[11] Q. Cappart, D. Chételat, E. B. Khalil, A. Lodi, C. Morris, and P. Velickovic. Combinatorial optimization and reasoning with graph neural networks. *J. Mach. Learn. Res.*, 24:130:1–130:61, 2023.

[12] E. M. Clarke, A. Biere, R. Raimi, and Y. Zhu. Bounded model checking using satisfiability solving. *Formal Methods Syst. Des.*, 19(1):7–34, 2001.

[13] S. A. Cook. The complexity of theorem-proving procedures. In M. A. Harrison, R. B. Banerji, and J. D. Ullman, editors, *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA*, pages 151–158. ACM, 1971.

[14] M. Cosler, F. Schmitt, C. Hahn, and B. Finkbeiner. Iterative circuit repair against formal specifications. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[15] A. Darbari, B. Fischer, and J. Marques-Silva. Industrial-strength certified SAT solving through verified SAT proof checking. In A. Cavalcanti, D. Déharbe, M. Gaudel, and J. Woodcock, editors, *Theoretical Aspects of Computing - ICTAC 2010, 7th International Colloquium, Natal, Rio Grande do Norte, Brazil, September 1-3, 2010. Proceedings*, volume 6255 of *Lecture Notes in Computer Science*, pages 260–274. Springer, 2010.

[16] M. Davis and H. Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, 1960.

[17] L. M. de Moura and N. S. Bjørner. Satisfiability modulo theories: introduction and applications. *Commun. ACM*, 54(9):69–77, 2011.

[18] V. Firoiu, E. Aygün, A. Anand, Z. Ahmed, X. Glorot, L. Orseau, L. M. Zhang, D. Precup, and S. Mourad. Training a first-order theorem prover from synthetic data. *CoRR*, abs/2103.03798, 2021.

 https://doi.org/10.52202/079017-1532

[19] E. First, M. N. Rabe, T. Ringer, and Y. Brun. Baldur: Whole-proof generation and repair with large language models. *CoRR*, abs/2303.04910, 2023.

[20] E. I. Goldberg and Y. Novikov. Verification of proofs of unsatisfiability for CNF formulas. In *2003 Design, Automation and Test in Europe Conference and Exposition (DATE 2003), 3-7 March 2003, Munich, Germany*, pages 10886–10891. IEEE Computer Society, 2003.

[21] C. Hahn, F. Schmitt, J. U. Kreber, M. N. Rabe, and B. Finkbeiner. Teaching temporal logics to neural networks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[22] M. Heule, W. A. H. Jr., and N. Wetzler. Bridging the gap between easy generation and efficient verification of unsatisfiability proofs. *Softw. Test. Verification Reliab.*, 24(8):593–607, 2014.

[23] M. J. H. Heule. Proofs of unsatisfiability. In A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors, *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 635–668. IOS Press, 2021.

[24] M. J. H. Heule and O. Kullmann. The science of brute force. *Commun. ACM*, 60(8):70–79, 2017.

[25] G. Irving, C. Szegedy, A. A. Alemi, N. Eén, F. Chollet, and J. Urban. Deepmath - deep sequence models for premise selection. In D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2235–2243, 2016.

[26] M. Järvisalo, M. Heule, and A. Biere. Inprocessing rules. In B. Gramlich, D. Miller, and U. Sattler, editors, *Automated Reasoning - 6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26-29, 2012. Proceedings*, volume 7364 of *Lecture Notes in Computer Science*, pages 355–370. Springer, 2012.

[27] H. A. Kautz and B. Selman. Planning as satisfiability. In B. Neumann, editor, *10th European Conference on Artificial Intelligence, ECAI 92, Vienna, Austria, August 3-7, 1992. Proceedings*, pages 359–363. John Wiley and Sons, 1992.

[28] E. B. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song. Learning combinatorial optimization algorithms over graphs. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6348–6358, 2017.

[29] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[30] W. Kool, H. van Hoof, and M. Welling. Attention, learn to solve routing problems! In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[31] P. Lammich. Efficient verified (UN)SAT certificate checking. *J. Autom. Reason.*, 64(3):513–532, 2020.

[32] G. Lederman, M. N. Rabe, S. Seshia, and E. A. Lee. 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.

[33] S. M. Loos, G. Irving, C. Szegedy, and C. Kaliszyk. Deep network guided proof search. In T. Eiter and D. Sands, editors, *LPAR-21, 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Maun, Botswana, May 7-12, 2017*, volume 46 of *EPiC Series in Computing*, pages 85–105. EasyChair, 2017.

[34] J. Marques-Silva, I. Lynce, and S. Malik. Conflict-driven clause learning SAT solvers. In A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors, *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 133–182. IOS Press, 2021.

[35] M. Mikula, S. Antoniak, S. Tworkowski, A. Q. Jiang, J. P. Zhou, C. Szegedy, L. Kucinski, P. Milos, and Y. Wu. Magnushammer: A transformer-based approach to premise selection. *CoRR*, abs/2303.04488, 2023.

[36] E. Ozolins, K. Freivalds, A. Draguns, E. Gaile, R. Zakovskis, and S. Kozlovics. Goal-aware neural SAT solver. In *International Joint Conference on Neural Networks, IJCNN 2022, Padua, Italy, July 18-23, 2022*, pages 1–8. IEEE, 2022.

[37] A. Paliwal, S. M. Loos, M. N. Rabe, K. Bansal, and C. Szegedy. Graph representations for higher-order logic and theorem proving. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 2967–2974. AAAI Press, 2020.

[38] R. Pascanu, T. Mikolov, and Y. Bengio. Understanding the exploding gradient problem. *CoRR, abs/1211.5063*, 2(417):1, 2012.

[39] S. Polu, J. M. Han, K. Zheng, M. Baksys, I. Babuschkin, and I. Sutskever. Formal mathematics statement curriculum learning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[40] S. Polu and I. Sutskever. Generative language modeling for automated theorem proving. *CoRR*, abs/2009.03393, 2020.

[41] F. Schmitt, C. Hahn, M. N. Rabe, and B. Finkbeiner. Neural circuit synthesis from specification patterns. In M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 15408–15420, 2021.

[42] D. Selsam and N. S. Bjørner. Guiding high-performance SAT solvers with unsat-core predictions. In M. Janota and I. Lynce, editors, *Theory and Applications of Satisfiability Testing - SAT 2019 - 22nd International Conference, SAT 2019, Lisbon, Portugal, July 9-12, 2019, Proceedings*, volume 11628 of *Lecture Notes in Computer Science*, pages 336–353. Springer, 2019.

[43] D. Selsam, M. Lamm, B. Bünz, P. Liang, L. de Moura, and D. L. Dill. Learning a SAT solver from single-bit supervision. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[44] L. Sun, D. Gérault, A. Benamira, and T. Peyrin. Neurogift: Using a machine learning based sat solver for cryptanalysis. In S. Dolev, V. Kolesnikov, S. Lodha, and G. Weiss, editors, *Cyber Security Cryptography and Machine Learning - Fourth International Symposium, CSCML 2020, Be'er Sheva, Israel, July 2-3, 2020, Proceedings*, volume 12161 of *Lecture Notes in Computer Science*, pages 62–84. Springer, 2020.

[45] G. S. Tseitin. On the complexity of derivation in propositional calculus. *Automation of reasoning: 2: Classical papers on computational logic 1967–1970*, pages 466–483, 1983.

[46] O. Vinyals, M. Fortunato, and N. Jaitly. Pointer networks. *Advances in neural information processing systems*, 28, 2015.

[47] Y. Vizel, G. Weissenbacher, and S. Malik. Boolean satisfiability solvers and their applications in model checking. *Proc. IEEE*, 103(11):2021–2035, 2015.

[48] M. Wang, Y. Tang, J. Wang, and J. Deng. Premise selection for theorem proving by deep graph embedding. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 2786–2796, 2017.

[49] W. Wang, Y. Hu, M. Tiwari, S. Khurshid, K. L. McMillan, and R. Miikkulainen. Neuroback: Improving CDCL SAT solving using graph neural networks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

 https://doi.org/10.52202/079017-1532

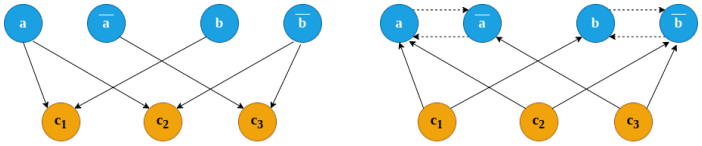
[50] N. Wetzler, M. Heule, and W. A. H. Jr. Drat-trim: Efficient checking and trimming using expressive clausal proofs. In C. Sinz and U. Egly, editors, *Theory and Applications of Satisfiability Testing - SAT 2014 - 17th International Conference, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings*, volume 8561 of *Lecture Notes in Computer Science*, pages 422–429. Springer, 2014.

[51] K. Yang and J. Deng. Learning to prove theorems via interacting with proof assistants. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6984–6994. PMLR, 2019.

[52] L. Zhang and S. Malik. Validating SAT solvers using an independent resolution-based checker: Practical implementations and other applications. In *2003 Design, Automation and Test in Europe Conference and Exposition (DATE 2003), 3-7 March 2003, Munich, Germany*, pages 10880–10885. IEEE Computer Society, 2003.

Appendix

A NeuroSAT Formula Graph Construction



(a) Literal-to-Clause Phase (b) Clause-to-Literal Phase

Figure 6: Two-phase message-passing round on NeuroSAT formula graph.

NeuroSAT-style formula graphs have two designated node types: clause nodes connected to the literal nodes corresponding to their constituent literals [43]. For example, in Figure 6, the clause contents are as follows: $c_1 = (a \vee b)$, $c_2 = (a \vee \bar{b})$, $c_3 = (\bar{a} \vee \bar{b})$. Each message-passing round involves two exchange phases: (1) Literal-to-Clause, and (2) Clause-to-Literal (and implicitly Literal-to-Complement). This construction is particularly efficient as it allows the message-passing protocol to cover the entire graph connectivity in at most $|V| + 1$ rounds where V is the set of variables in the formula.

B Clause Connectivity Under Static Embeddings

In Section 4.2, we stated that under static embeddings for a derived clause, as the embedder creates its embedding, it only updates the representations of the variables involved in it – leaving other clause embeddings intact. This might present a problem for Full-Attn where the attention grid contains all clauses including disconnected[1] pairs. An example of such a pair would be two derived clauses that do not share a variable. This could potentially lower the efficacy of the Full-Attn mechanism as it tries to match clauses that are unaware of each other. Interestingly, despite being a relaxation on Full-Attn, Anch-Attn has a distinct edge over Full-Attn under static embeddings in form of the following property:

Lemma B.1. *Clauses in the variable-anchored attention grid of Anch-Attn are guaranteed to be connected under both static and dynamic embeddings.*

Proof. Let v be a variable in the input formula, and the set of clauses of a v-anchored attention grid be A. We show that we always have at least one clause $A_i \in A$ that reaches all other clauses in A on the formula graph. We make two case distinctions:

Case 1: All clauses in A are input clauses (in the original formula). Here, the lemma follows trivially since all these clause were connected during the input-phase message-passing protocol as they share at least one variable v.

Case 2: A contains derived clauses. Let A_i be the most recently derived clause in A. Since A_i shares variable v with all other clauses in A, then A_i would be connected to them all during the derivation-phase message-passing protocol immediately after A_i was derived. This is because A_i receives a message from V (under both static and dynamic embeddings) containing information about all other clauses containing v, which is precisely $A \setminus \{A_i\}$. Therefore, the lemma holds. □

[1]We use the terms ***connected*** and ***disconnected*** here to refer to the fact of whether two nodes have exchanged messages (in either direction) or not, respectively.

https://doi.org/10.52202/079017-1532

C Teacher Proof Reduction

One rather interesting observation on Table 5 is that the model appears to be marginally better at producing shorter proofs for unseen (test) formulas than for training formulas. While we would normally expect the opposite, a fair speculation would be that the trained model was teacher-forced to match teacher proofs during training over multiple epochs while the same does not hold for unseen formulas where the bias towards teacher behavior is significantly lower. Definitively confirming this would require a more in-depth investigation.

Table 5: Teacher proof reduction statistics of non-bootstrapped model trained on unreduced $\mathbf{SR}(U(10, 40))$ dataset. Note that all rows, except for Total Reduction, are computed over the *reduced portion* of the dataset, i.e., the proofs that were successfully shortened by NeuRes.

(%)	TRAIN	TEST
PROOFS REDUCED	17.82	18.29
MAX. REDUCTION	86.11	76.4
AVG. REDUCTION	23.55	23.65
TOTAL REDUCTION	3.07	3.15

D Runtime Comparison with Traditional Solver

In the following table, we compare the average runtimes of our top-1 Full-Attn, top-3 Full-Attn, and the traditional solver we used as a teacher (BooleForce) on our main $\mathbf{SR}(40)$ test dataset. For both Full-Attn models, we use our Python prototype implementation; for BooleForce, we use an official C implementation.

Table 6: Average time (ms) to solve an instance by neural model vs. teacher solver.

SOLVER	SAT (MS)	UNSAT (MS)	TOTAL (MS)
FULL-ATTN TOP-1	2.3	88	45.15
FULL-ATTN TOP-3	3	54.4	28.7
BOOLEFORCE	4	5	4.5

E Model Size Comparison with NeuroSAT

In terms of the model architecture, both NeuRes and NeuroSAT models can be broken down to:

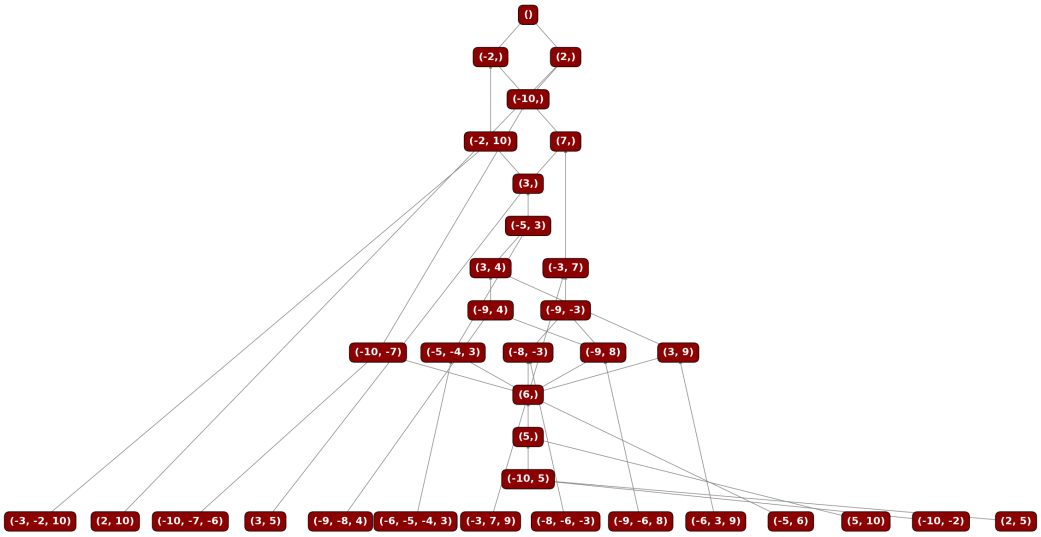
- **Embedding/Representation Network:** for both models, this network is an LSTM-based GNN that embeds the formula graph by message-passing. We use the exact same architecture and model size to ensure that our improved representations are a result of our fully certificate-based learning objective as opposed to a tweak in the model architecture. This GNN has $429,824$ parameters in total.
- **Downstream Networks:** NeuroSAT: uses a 3-layer MLP applied on the literal embeddings (width $= 128$) to extract the literal votes to predict if the formula is satisfiable or not. This MLP has $128 \times 128 \times 3 = 49,152$ parameters. NeuRes (Full-Attn): uses an attention module to select clause pairs. This attention network is composed of two 1-layer MLPs for the query and key transformations on the clauses embeddings (width $= 128$). The whole attention module has $128 \times 128 \times 2 = 32,768$ parameters. To decode the variable assignments, NeuRes uses a 2-layer scalar MLP with $128 \times 128 + 128 = 16,512$ parameters

Total NeuroSAT size $= 429,824 + 49,152 = 478,976$ parameters

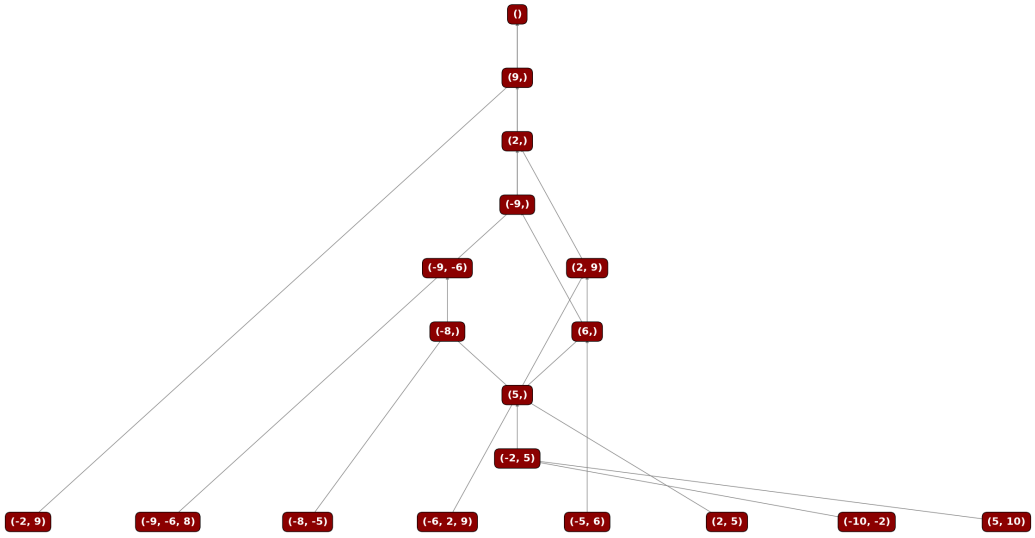
Total NeuRes size $= 429,824 + 49,280 = 479,104$ parameters

All in all, NeuRes only learns 128 more parameters.

https://doi.org/10.52202/079017-1532



(a) Teacher Proof



(b) NeuRes Proof

Figure 7: Teacher Proof Reduction Example

https://doi.org/10.52202/079017-1532

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We included a contributions section in the introduction which refers to the corresponding sections in the paper.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have highlighted the limitations with respect to industrial-grade SAT solving in the conclusion of the paper (Section 10).

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: The paper does not include any formal theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We reference the data generation method in Section 5.1. A detailed mathematical description of the architectures is given in Section 4. Optimization details and hyperparameters are disclosed in Section 5.2 and Section 5.3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

https://doi.org/10.52202/079017-1532

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Data and code are publicly available through the link provided in the paper.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Training and test splits are specified at the beginning of Section 6 and Section 7. Optimization and hyperparameters are detailed in Section 5.2 and Section 5.3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Due to the computational budget, errors bars are not reported.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The compute resources are reported in Section 5.3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: No harms were caused by the research process nor are harmful consequences expected from the propositional logic problems considered in this work.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: As discussed in the conclusion, a potential application of this work is to build hybrid SAT solvers. Such solvers are used in a variety of applications to improve the reliability of software and hardware, and hence this work may have positive societal impact in this area. Since the proposed models back their verdicts by easily verifiable certificates, there is little risk of harm since incorrect results can be detected easily, unlike in, e.g., mere satisfiability prediction.

    Guidelines:

https://doi.org/10.52202/079017-1532

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA] .

    Justification: We do not recognize any risk of misuse of the datasets and models presented in this work.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [NA]

    Justification: No existing assets were used.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We will make new assets publically available after the double-blind review ended.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

https://doi.org/10.52202/079017-1532

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.