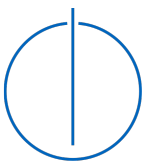
SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY – INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Guided Research Project Report

Adversarial Training for Neural Combinatorial Solvers

Oriol Aranda Llorens



SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY – INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Guided Research Project Report

Adversarial Training for Neural Combinatorial Solvers

Adverses Training für neurale kombinatorische Löser

Author:	Oriol Aranda Llorens
Supervisor:	Prof. Dr. Stephan Günnemann
Advisor:	Johanna Sommer
Submission Date:	April 21st 2023

I confirm that this guided research project report is my own work and I have documented all sources and material used.

Munich, April 21st 2023 Oriol Aranda Llorens

Abstract

Combinatorial optimization problems present a significant challenge due to the exponential number of possibilities, rendering traditional algorithms inefficient. Leveraging the significant advances in deep learning, specially Graph Neural Networks (GNNs), neural combinatorial solvers have emerged as a promising alternative for solving these problems, but their limited generalization and extrapolation capabilities make them less competitive than classical solvers. Adversarial robustness has become a crucial area of research for neural combinatorial solvers, which have been shown to be highly non-robust against adversarial attacks on perturbed instances. These attacks highlight the model's limitations and identify regions that are difficult for the model and not covered by the training data. In this study, we propose a novel adversarial training method using the Fast Gradient Sign Method (FGSM) to enhance the robustness and generalization of neural combinatorial solvers for the SAT problem. We demonstrate the effectiveness of our approach on the CircuitSAT problem and observe promising results in improving the solvers' generalization and robustness. Our proposed method holds great potential for advancing the development of more robust and generalizable neural combinatorial solvers, representing a significant step towards addressing combinatorial optimization problems.

Contents

1 Introduction

Computer science began to be established as a discipline in the 1950s, and has been developing until today with the aim of solving problems encountered in our daily lives. Many of the most interesting and useful ones are related with finding an optimal object from a finite set of objects, such as the Travelling Salesperson Problem (TSP), the Minimum Spanning Tree (MST) or the Boolean Satisfiability Problem (SAT) [PS98]. Those are so-called combinatorial optimization problems and they have an extremely wide number of applications [PS98]. Generally, in all these problems we are searching for a solution from among an exponential population of possibilities, and they could in principle be solved in exponential time by checking through all candidate solutions. Suppose having a TSP instance with 20 cities, and checking a possible solution takes 1 second. Then, since there are factorial possibilities, our algorithm's running time is $20!$ s $\approx 2.43 \times 10^{18}$ s, approximately 77.1 billion years; all but useless in practice.

The last decades, theorists have made great efforts to understand, classify and find efficient algorithms for them. And indeed, not all the combinatorial optimization problems have the same difficulty; efficient algorithms have been successfully developed for a number of them. Nevertheless, the fastest algorithms we know for the rest, which are the hardest to solve and generally the most useful, are all exponential, which is not substantially better than an exhaustive search [DPV08]. These hard problems are known to be NP-complete, which means that no polynomial time algorithms exist to solve them efficiently. Thus, solving them in a reasonable amount of time, requires to resort to approximation algorithms, heuristics or machine learning.

Recently, there have been some attempts to solve combinatorial problems with deep learning [BLP21], leveraging the growth of the field. Specifically, as many of these problems can be represented as a graph, Graph Neural Networks are most often used to leverage the graph structure of the problem and introduce useful invariances to the problem modelling. While there have been first successes, current models still struggle with generalization to large problem instances or extrapolation to and interpolation between instances drawn from different distributions [Gei+22; YGS20]. So, in general, these so-called neural solvers models are not competitive with traditional solvers which have been cleverly crafted during decades.

This is in part because, due to the large problem space, even for moderate problem sizes, it is intractable to sample it densely. Actually, generating the training data for these neural solvers is known to be either incomplete but efficient, or complete but inefficient, which is problematic [YGS20]. In the first case, we can construct instances with known labels efficiently, but then only a subset of the problem space is covered. Alternatively, in the second case, we can sample different instances at random and approximate the label with a state of the art solver, which can reflect better the problem space, but can be prohibitively expensive. Additionally, it is concerning that often the same incomplete data generator is used for training and evaluation, leading to too optimistic results.

The latest has sparked interest in studying adversarial robustness for neural combinatorial solvers, which more realistically evaluates a model's generalization abilities [Gei+22]. Indeed, interestingly, adversarial robustness is a desirable property in this context, since, in contrast to general learning tasks, here we do not have an accuracy robustness trade-off. That is, there exists a model with high accuracy and high robustness. Through adversarial attacks, i.e., perturbed instances, it has been shown that current neural solvers are highly non-robust. These attacks reveal the regions that are both difficult for the model and not covered by training samples. The idea is to use these hard model-specific samples to train a model with improved capabilities.

This project proposes the use of adversarial training to improve the generalization and robustness of neural combinatorial solvers, as well as to facilitate learning with costly generated data. Specifically, we apply the Fast Gradient Sign Method (FGSM) [GSS14] to adversarially train the neural solver CircuitSAT [AMW19] for the SAT problem, demonstrating improved generalization and robustness. Moreover, our approach contributes to the development of more efficient data generation procedures for SAT. Overall, our proposed method has the potential to contribute to the development of more robust and generalizable neural combinatorial solvers. Our hope for the future is that neural combinatorial solvers will offer a significant advantage over classical solvers and mark a significant step forward in addressing combinatorial optimization problems.

2 Background

2.1 Boolean Satisfiability Problem (SAT)

The Boolean satisfiability problem (SAT) is a well-studied combinatorial optimization problem in computer science [AB09]. The problem involves determining the satisfiability of a given Boolean formula by assigning Boolean values to its variables such that the formula evaluates to true.

Formally, a Boolean formula ϕ in conjunctive normal form (CNF) with n variables $x_1, x_2, \ldots, x_n$ and m clauses $C_1, C_2, \ldots, C_m$, is given by:

$$\phi = \bigwedge_{i=1}^{m} C_i,$$

where each clause C_i is a disjunction of literals (variables or their negations):

$$C_i = l_{i,1} \vee l_{i,2} \vee \cdots \vee l_{i,k_i},$$

and $l_{i,j}$ is either a variable x_j or its negation $\neg x_j$.

Thus, the SAT problem can be formally stated as follows: Given a Boolean formula ϕ in CNF form with n variables and m clauses, determine whether there exists an assignment of Boolean values to the variables $x_1, x_2, \ldots, x_n$ that satisfies all the clauses $C_1, C_2, \ldots, C_m$ in ϕ. If such an assignment exists, then ϕ is satisfiable; otherwise, ϕ is unsatisfiable. The SAT problem arises in several forms, and the most relevant to this work is the k-SAT problem. The k-SAT problem is a specific instance of the underlying problem distribution of SAT, which encompasses various types of SAT problems with different distributions of clauses and literals. In k-SAT, the problem is characterized by a fixed number of literals $k_i = k$ for all clauses C_i.

The SAT problem has numerous practical applications in computer science, including hardware and software verification, automated theorem proving, and planning. Moreover, it is an essential problem in computational complexity theory, as it is one of the canonical NP-complete problems [GJ79]. This means that the SAT problem is believed

to be computationally intractable in the worst case, and finding a polynomial-time algorithm to solve it would imply that $\mathsf{P} = \mathsf{NP}$, solving the P vs NP problem, the major unsolved problem in theoretical computer science [Coo71; Lev73] and one of the seven Millennium Prize Problems in mathematics.

Despite its theoretical difficulty, researchers have made substantial progress in solving practical instances of SAT. This progress can be attributed to the development of highly efficient algorithms, heuristics, and techniques, such as the DPLL algorithm [DP60; DLL62], clause learning (CDCL) [MS99; BS97], and conflict analysis. These techniques have led to the creation of highly effective SAT solvers capable of handling instances with millions of variables and clauses.

2.2 Neural Combinatorial Solvers

Neural combinatorial solvers (NCS) represent a burgeoning field of research focused on utilizing deep learning to solve complex combinatorial optimization problems. These problems require selecting the best solution from a finite set of possible solutions, where the search space is discrete and expansive [PS98]. Formally, we can define a (unique) optimal solution

$$y = \arg \min_{y' \in g(x)} c(x, y')$$

where $x \in \mathbb{X}$ is the problem instance, $g(x) = \mathbb{Y}$ the finite set of feasible solutions for x, and c the cost function. Throughout this work, we assume the existence of a single optimal solution. However, it is important to acknowledge that some problems may have multiple optimal solutions, resulting in a set of feasible solutions that meet the optimization criteria.

A neural solver f_θ learns a mapping $f_\theta : \mathbb{X} \to \mathbb{Y}$ that approximates the optimal solution. In a supervised learning setting, the parameters θ are optimized with respect a loss ℓ over a finite set of training samples (x, y). Formally:

$$\min_\theta \ell(f_\theta(x), y)$$

In the context of SAT, during the last years, several NCS models have been proposed with promising results. Among these models, the use of graph neural networks (GNNs) has shown significant promise, since they can leverage the graph structure of the previously encoded formulas, capturing important relationships between variables, and clauses. Such models, e.g., NeuroSAT [Sel+18] and CircuitSAT [AMW19], have demonstrated state-of-the-art performance on solving SAT using end-to-end models. See

Bengio et al. [BLP21] for a quite recent survey. Several prior works have concentrated on improving variable branching heuristics for classical solvers [Wan+21; SB19]. Other research has focused on other end-to-end architectures, including transformer-based approaches and reinforcement learning techniques [Li+22; Shi+22].

CircuitSAT proposed by Amizadeh et al. [AMW19], represents the CNF formula ϕ as a directed acyclic graph (DAG), denoted by x_ϕ, known as a Boolean circuit. In such representation, each variable, its negation, the clauses and the final AND are represented as nodes. There is a directed edge between a variable and its negation; a variable and the clause where it appears; and, a clause and the root (AND). To simplify, the model used in CircuitSAT consist of two so-called networks. Firstly, the solver network, which produces an assignment for each variable of the formula ϕ. Formally, $\mathcal{F}_\theta(x_\phi) = \mathcal{C}_\alpha(\mathcal{E}_\beta(x_\phi))$, where $\theta = (\alpha, \beta)$, the embedding function $\mathcal{E}_\beta$ is a Deep-Gated DAG Recurrent Neural Network (DG-DAGRNN), and the classification function $\mathcal{C}_\alpha$ is a multi-layer neural network. Secondly, the evaluator network $\mathcal{R}_\phi$, which evaluates the formula ϕ given an assignment for all its variables. Putting all pieces together, the satisfiability function:

$$\mathcal{S}_\theta(x_\phi) = \mathcal{R}_\phi(\mathcal{F}_\theta(x_\phi))$$

is trained using the smooth step function:

$$\ell(s) = \frac{(1-s)^k}{(1-s)^k + s^k},$$

where $s = S_\theta(x_\phi)$ and $k \geq 1$ is a constant. The loss encourages the solver network to find a feasible assignment if the input formula is satisfiable. As a remark, the model is trained just with satisfiable samples.

Regarding data generation, the authors of NeuroSAT by Salsam et al. [Sel+18] use an iterative procedure to construct each formula ϕ randomly. Given the number of variables n, the process involves adding a randomly generated clause C_i to the formula iteratively until it becomes unsatisfiable, with $1 \leq i \leq m$, m being the number of clauses. For each clause, the number of variables k_i is determined by drawing from the distribution $k_i \sim 1 + Ber(0.7) + Geo(0.4)$. However, the process is computationally expensive for large n since a classical SAT solver is employed at each step to verify the final condition.

2.3 Adversarial Training

In recent years, adversarial training has emerged as a powerful technique for improving the robustness of machine learning models [Sze+13]. The approach involves training a model on a dataset that includes adversarial examples, which are inputs designed to cause the model to make errors. Thus, adversarial training can be understood as a powerful data augmentation using hard model-specific instances.

Specifically, let us consider a classifier f_θ that maps an input x to a label y, and an adversarial example $\tilde{x}$ that is constructed by applying a perturbation ρ to the original input, such that $\tilde{x} = x + \rho$. The goal of adversarial training is to optimize the model's parameters θ to minimize the worst-case loss over a set of adversarial examples:

$$\min_\theta \max_\rho \ell(f_\theta(x + \rho), y)$$

where ℓ is the loss function, and ρ the perturbation parameters.

One of the most common approaches to learning the best perturbation is to use gradient-based methods. These methods involve computing the gradient of the loss function with respect to the input, and then using this gradient to perturb the input in a way that maximizes the loss. The Fast Gradient Sign Method (FGSM) [GSS14] involves adding a small perturbation ϵ in the direction of the sign of the gradient. Specifically, given an input sample x, the label y, a loss function $J(\theta, x, y)$ that depends on the model parameters θ, and a small scalar ϵ, the perturbed input $\tilde{x}$ is computed as follows:

$$\tilde{x} = x + \epsilon \cdot \text{sign}(\nabla_x J(\theta, x, y)) \tag{2.1}$$

where $\text{sign}(\cdot)$ returns the sign of its argument.

The Projected Gradient Descent (PGD) [Mąd+17] method extends FGSM by iteratively applying FGSM with small step sizes and projecting the resulting perturbation back onto a valid input region. Specifically, given an input sample x, a label y, a loss function $J(\theta, x, y)$, a small scalar ϵ, a step size α, and a maximum number of iterations T, the perturbed input $\tilde{x}$ is computed as follows:

$$\begin{aligned}
&\tilde{x}_0 = x \\
&\tilde{x}_{t+1} = \text{proj}_{x,\epsilon}(\tilde{x}_t + \alpha \cdot \text{sign}(\nabla_{\tilde{x}_t} J(\theta, \tilde{x}_t, y)))\Big)
\end{aligned}$$

where $\text{proj}_{x,\epsilon}(\cdot)$ is a function that projects its argument onto the ϵ-ball around the original value x. After T iterations, the perturbed instance $\tilde{x}$ is set to $\tilde{x}_T$. In some

variations of the method, the gradient is used instead of the sign.

In contrast to other domains, perturbing instances in SAT can change the label, making it a challenging task to generate adversarial examples. However, we can still leverage the invariances of the SAT problem to create perturbed instances while preserving the label [Gei+22]. Specifically, we can add or remove literals from a satisfiable formula ϕ, as long as we ensure that at least one true literal remains in each clause. This is achieved through a projection operator, which selects one true literal from each clause and ensures that it is not removed. By using this approach, we can generate perturbed hard instances that maintain satisfiability.

For example, consider the satisfiable formula $\phi = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_3)$. To create a perturbed instance while preserving the label, we can apply the projection operator to select one true literal from each clause and ensure that it is not removed. In this case, we can select x_1 and x_3 as the true literals for the first and second clause, respectively. We can then arbitrarily add or remove other literals as long as we maintain at least one true literal in each clause. For instance, we can add the literals x_4 and $\neg x_5$, and remove the literal $\neg x_1$, to obtain the formula $\phi' = (x_1 \vee x_2 \vee x_4) \wedge (x_3 \vee \neg x_5)$, which is still satisfiable.

3 Methodology

This section outlines the methodology used to develop and test our proposed approach for improving the generalization and robustness of the neural combinatorial solver CircuitSAT to solve the Boolean satisfiability (SAT) problem. Our approach consists of two key components: (1) a data generation procedure that constructs the problem instances randomly from a solution to produce data more efficiently, and (2) an adversarial training procedure that employs the Fast Gradient Sign Method (FGSM) to create the perturbed training samples.

3.1 Data Generation

Building on the data generation procedure proposed in NeuroSAT by Selsam et al. [Sel+18], we propose a novel data generation method for SAT and k-SAT that is more efficient and does not rely on classical solvers. Our method involves randomly selecting a solution, denoted by y, and iteratively generating clauses C_i with k_i variables each, $1 \leq i \leq m$. For SAT, the value of k_i is sampled from the distribution $k_i \sim 1 + \text{Ber}(0.7) + \text{Geo}(0.4)$, where Ber and Geo denote the Bernoulli and geometric distributions, respectively. The number of clauses m is drawn based on statistics obtained from a dataset of 300,000 samples previously generated using the former method. Specifically, we set $m \sim N(\mu, \sigma^2)$, where μ and σ are the sample mean and standard deviation of the number of clauses in the 300,000 samples dataset given some n.

For k-SAT, the value of k is fixed for each clause, and we use the phase transition factor $\alpha_k = m/n$ to generate hard SAT instances. The phase transition factor relates the number of variables n, the number of clauses m, and the number of variables per clause k and is used to estimate the transition point between satisfiable and unsatisfiable instances for a given k, thus the hardest ones. We obtain the phase transition factors from the work of Mertens et al. [MMZ06].

The data generator for SAT and k-SAT is described above and the pseudocode is presented in Algorithm 1 for clarity and conciseness. Note that the function $sample(c, n)$, sample n values from the collection c.

Algorithm 1: Generate SAT or k-SAT instance

Input: n: number of variables, k: number of literals per clause
Output: ϕ: a SAT or k-SAT instance

$x \leftarrow \{x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n\}$
$y \leftarrow sample(x,\ n)$
if *SAT* **then**
  $m \leftarrow \mathcal{N}(\mu,\ \sigma^2)$ `// from the 300,000 samples`
else
  $m \leftarrow \alpha_k \cdot n$ `// where` α_k `is the phase transition factor for k`
$\phi \leftarrow \varnothing$
for $i = 1$ *to* m **do**
  if *SAT* **then**
    $k_i \leftarrow 1 + \mathrm{Ber}(0.7) + \mathrm{Geo}(0.4)$
  else
    $k_i \leftarrow k$
  $l_1 \leftarrow sample(y,\ 1),\quad l_s \leftarrow sample(x,\ k_i - 1)$
  $c_i \leftarrow \{l_1\} \cup l_s$ `// random clause with` k_i `variables`
  $\phi \leftarrow \phi \cup \{c_i\}$
return ϕ

3.2 Adversarial Training

We propose an extension of the Fast Gradient Sign Method (FGSM) to the domain of neural combinatorial solvers, specifically applied to the CircuitSAT model. The procedure involves encoding the input formula ϕ as a graph, following the methodology described in the original paper. We then utilize the adjacency matrix of the encoded formula x_ϕ in combination with a perturbation matrix ρ, which is learned through optimization using the Equation 2.1. The idea of the perturbation matrix ρ is similar in nature to the Projective Gradient Descent (PGD) technique used by Geisler et al. [Gei+22], which was applied to the NeuroSAT model. Upon optimization, the perturbation matrix ρ is applied to the adjacency matrix, resulting in the addition and deletion of some literals.

Indeed, the perturbation matrix ρ used in our approach can be interpreted as the probability of each variable being added or deleted from a clause. Specifically, ρ_{ij} represents the probability of the j-th variable in clause i being flipped, where a value of 0 indicates that the variable will not be flipped and a value of 1 indicates that the

variable will be flipped. By optimizing ρ using the FGSM, we can determine the optimal values of the probabilities for each variable in order to maximize the likelihood of finding a valid solution. Moreover, the ϵ parameter in FGSM determines the magnitude of the perturbation, as it scales the sign of the gradient, which in turn affects the probability of adding or deleting variables. Specifically, a larger ϵ value will result in a larger perturbation and a higher probability of modifying the adjacency matrix, while a smaller ϵ value will lead to a smaller perturbation and a lower probability of changing the matrix.

Gradient-based optimization methods rely on the computation of derivatives, i.e., information about how the function changes with small perturbations of its input variables. However, when the input variables are discrete, the concept of a derivative does not apply since the variables cannot be infinitesimally perturbed. To overcome this limitation, optimizing the perturbation matrix requires continuous values, which are then discretized using a Bernoulli probability distribution represented by the matrix ρ. To ensure that the perturbation matrix values remain within the 0-1 probability range, a clipping projection is applied. Additionally, to preserve satisfiability, also a satisfiability projection mask is used to ensure the preservation of at least one true literal for each clause. This results in a perturbed adjacency matrix with some literals added or removed based on the probabilities represented by ρ. The perturbed instance $\tilde{x}_\phi$ obtained can be used to adversarially train the CircuitSAT model.

In order to show the effectiveness of the proposed method for better generalization and robustness, our CircuitSAT model is trained with 300,000 instances generated on the fly by our data generation method, for two different sizes:

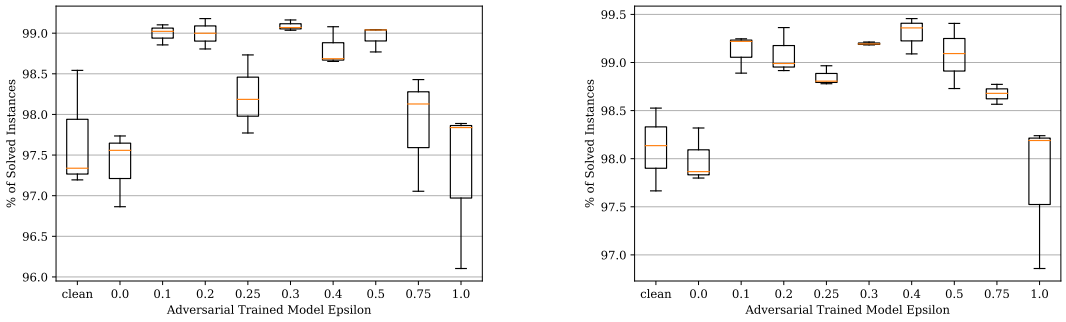
1. SAT-3-10, which is composed by formulas with 3 to 10 variables.
2. SAT-10-40, which contains formulas with 10 to 40 variables.

Firstly, both models were trained on clean instances and referred to as clean models, to demonstrate that adversarial training does not have a negative impact on training performance. Secondly, both models were adversarially trained using Fast Gradient Sign Method (FGSM) with different attack strengths $\epsilon \in \{0.1, 0.2, 0.25, 0.3, 0.4, 0.5, 0.75, 1.0\}$ depending on the experiment we were interested in. Throughout the rest of this study, we use the notation CSAT SAT-3-10 to refer to the version of CircuitSAT that was trained using the SAT-3-10 data, and similarly, CSAT SAT 10-40 refers to the version trained using the SAT 10-40 data.

4 Results

This section presents the results of the various experiments conducted in this study. The primary experiments conducted include (1) a comparison of the effectiveness of FGSM and PGD attacks on a clean model, (2) an evaluation of the generalization of adversarially trained models to larger instances, (3) an assessment of the generalization of adversarially trained models to out-of-distribution data, and (4) an examination of the robustness of the different adversarially trained models to FGSM attacks of varying strength. The experiments were conducted using three distinct seeded models, in conjunction with a validation set of approximately 30,000 samples, which accounted for roughly 10% of the total training samples.

Firstly, the evaluation of both trained models with unseen, i.e., larger instances is presented below in Figure 4.1, to show that adversarial training does not negatively impact training performance.



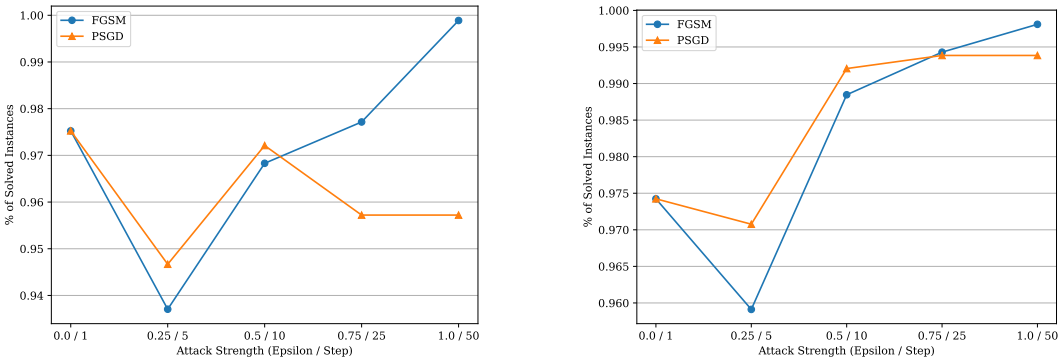
(a) CSAT SAT-3-10

(b) CSAT SAT-10-40

Figure 4.1: Percentage of Solved Instances

Effectiveness of FGSM Attack

In this study, we conducted a comparative analysis of the effectiveness of FGSM and PGD attacks against a clean trained CircuitSAT model, using different attack strengths represented by ϵ and the number of steps s, respectively, as illustrated in Figure Figure 4.2. The values of ϵ and s utilized in the study were $\epsilon \in \{0.0, 0.25, 0.5, 0.75, 1.0\}$ and $s \in \{1, 5, 10, 25, 50\}$. CSAT-SAT-3-10 is evaluated over SAT-10-40 and CSAT-SAT-10-40 over SAT-50-100. The results of our experiments clearly indicate that for small values of ϵ, FGSM attacks were more effective than PGD attacks, leading to a substantial drop in the model's performance.



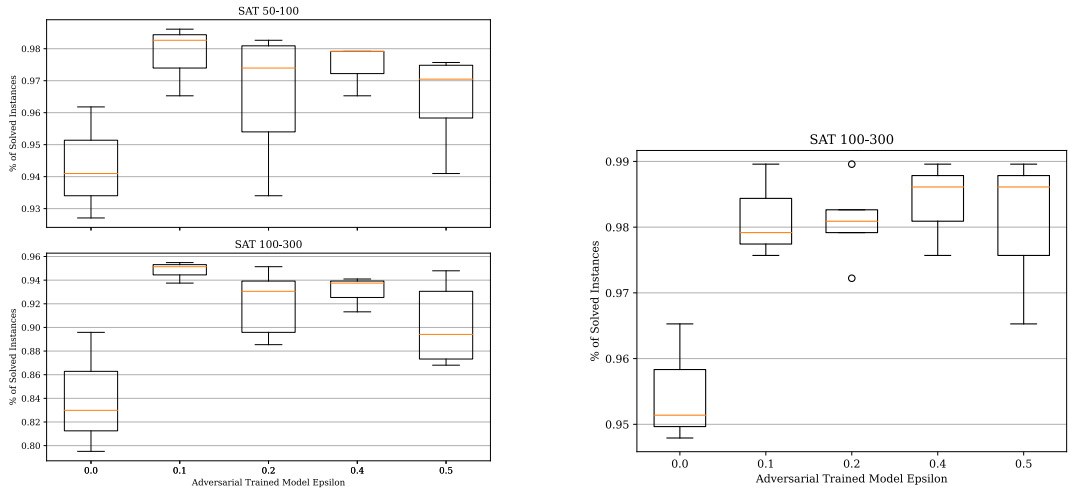
(a) CSAT SAT-3-10

(b) CSAT SAT-10-40

Figure 4.2: Effectiveness of FGSM and PGD attacks

Generalization to Larger Instances

As depicted in Figure 4.3, our experiments demonstrate that both models exhibited improved generalization on larger instances. Specifically, we evaluated the performance of adversarially trained CSAT-SAT-3-10 over SAT-50-100 and SAT-100-300, followed by the evaluation of adversarially trained CSAT-SAT-10-40 on SAT-100-300. In these experiments, we employed adversarially trained models with $\epsilon \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$, as we observed a degradation in performance above $\epsilon = 0.5$ with respect to the clean model. These results are particularly noteworthy since suggest that adversarial training can improve the ability of CircuitSAT models to generalize to larger problem instances, which could have important implications for the development of more robust and generalizable SAT neural combinatorial solvers.



(a) CSAT SAT-3-10

(b) CSAT SAT-10-40

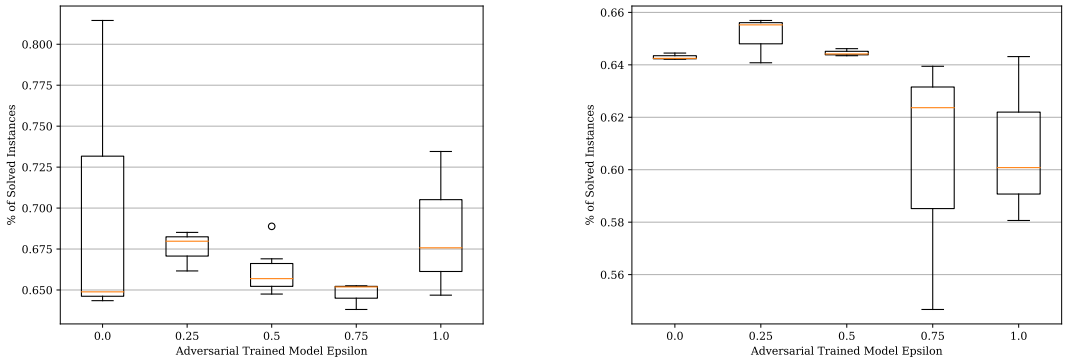
Figure 4.3: Generalization of CircuitSAT to larger instances

Generalization to Out-Of-Distribution Instances

As we hypothesized that adversarial training could improve the model's ability to generalize to out-of-distribution data, we designed experiments using k-SAT data to test this hypothesis. Specifically, we evaluated the performance of adversarially trained CSAT-SAT-3-10 over k-SAT-10-40, followed by the evaluation of adversarially trained CSAT-SAT-10-40 on k-SAT-50-100. In these experiments, we employed adversarially trained models with $\epsilon \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ and set $k \in \{3, 4, 5\}$. However, our experimental results showed in Figure 4.4 demonstrated that adversarial training did not yield any significant improvement in the model's generalization performance on out-of-distribution data.

Robustness of Adversarially Trained Model

The experimental results presented in this last study, Figure 4.5, highlight the critical importance of adversarial training for improving model robustness. In particular, our

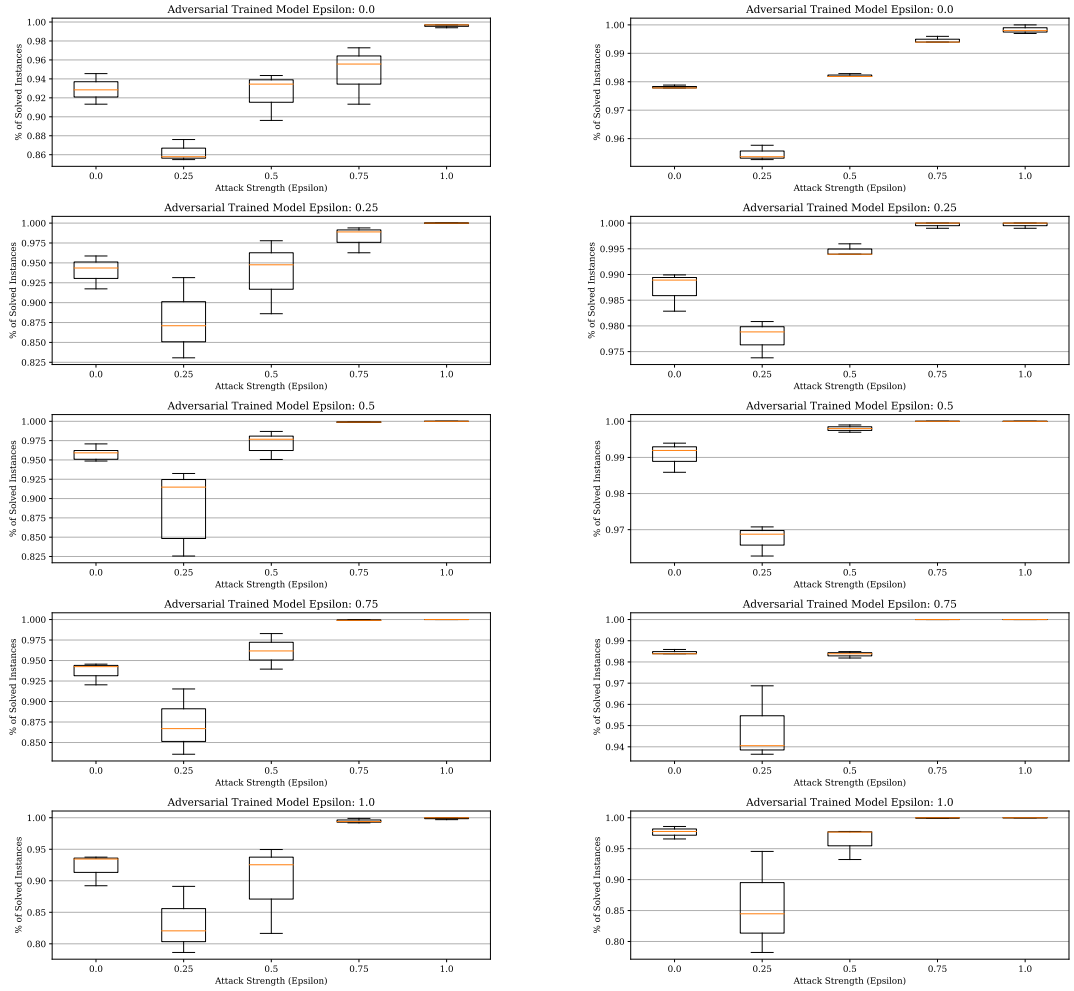


(a) CSAT SAT-3-10

(b) CSAT SAT-10-40

Figure 4.4: Generalization of CircuitSAT to out-of-distribution instances

findings provide strong empirical evidence that models trained on perturbed instances exhibit greater resilience against FGSM attacks.Specifically, we evaluated the robustness of adversarially trained CSAT-SAT-3-10 over perturbed instances of SAT-10-40, followed by the evaluation of adversarially trained CSAT-SAT-10-40 on perturbed instances of SAT-50-100. The models were adversarially trained with $\epsilon \in 0.25, 0.5, 0.75, 1$, and attacks were conducted using the same epsilons.



(a) CSAT SAT-3-10

(b) CSAT SAT-10-40

Figure 4.5: Robustness of CircuitSAT to different FGSM attack strengths

5 Conclusion

During the course of this work we have demonstrated the effectiveness of adversarial training for enhancing the generalization and robustness of neural combinatorial solvers, and improving the learning efficiency with costly generated data. We have shown that the Fastest Gradient Sign Method (FGSM) attack outperforms the Projective Gradient Descent (PGD) attack in terms of effectiveness. The proposed methodology leverages the Fast Gradient Sign Method (FGSM) for generating adversarial samples, and has demonstrated promising results in terms of improving the generalization of CircuitSAT on larger SAT problem instances.

Furthermore, the results obtained from our experiments demonstrate that the CircuitSAT model trained using adversarial training exhibits greater robustness against FGSM attacks. This highlights the significance of robustness against adversarial attacks, especially when dealing with combinatorial optimization problems, since all perturbed instances correspond to valid problem instances.

Moreover, we have presented a novel data generator for SAT and k-SAT that eliminates the need for classical solvers, thereby reducing the computational overhead and memory requirements. Our approach has shown to be particularly effective for larger instances with a large number of variables and clauses, where it significantly accelerates the training process without sacrificing the quality of the learned model.

We have shown how to enhance generalization and robustness with adversarial training for CircuitSAT. However, the question remains whether adversarial training can be applied to other models, or even to other combinatorial optimization problems. Therefore, future work could involve conducting a more in-depth study of other types of attacks, with more models, problem-specific data, or even other hard problems. This could further validate the effectiveness of adversarial training in enhancing the generalization and robustness of neural combinatorial solvers, and expand its applicability to other areas of combinatorial optimization.

List of Figures

Bibliography

[AB09] S. Arora and B. Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[AMW19] S. Amizadeh, S. Matusevych, and M. Weimer. "Learning to solve circuit-sat: An unsupervised differentiable approach." In: *International Conference on Learning Representations*. 2019.

[BLP21] Y. Bengio, A. Lodi, and A. Prouvost. "Machine Learning for Combinatorial Optimization: A Methodological Tour d'horizon." In: *Eur. J. Oper. Res.* 290.2 (2021), pp. 405–421. DOI: 10.1016/j.ejor.2020.07.063.

[BS97] R. J. Bayardo Jr and R. Schrag. "Using CSP look-back techniques to solve real-world SAT instances." In: *Aaai/iaai*. Citeseer. 1997, pp. 203–208.

[Coo71] S. A. Cook. "The complexity of theorem-proving procedures." In: *Proceedings of the third annual ACM symposium on Theory of computing*. 1971, pp. 151–158.

[DLL62] M. Davis, G. Logemann, and D. Loveland. "A machine program for theorem-proving." In: *Communications of the ACM* 5.7 (1962), pp. 394–397.

[DP60] M. Davis and H. Putnam. "A computing procedure for quantification theory." In: *Journal of the ACM (JACM)* 7.3 (1960), pp. 201–215.

[DPV08] S. Dasgupta, C. H. Papadimitriou, and U. V. Vazirani. *Algorithms*. McGraw-Hill Higher Education New York, 2008.

[Gei+22] S. Geisler, J. Sommer, J. Schuchardt, A. Bojchevski, and S. Günnemann. "Generalization of Neural Combinatorial Solvers Through the Lens of Adversarial Robustness." In: *International Conference on Learning Representations* (2022).

[GJ79] M. R. Garey and D. S. Johnson. *Computers and intractability*. Vol. 174. freeman San Francisco, 1979.

[GSS14] I. J. Goodfellow, J. Shlens, and C. Szegedy. "Explaining and harnessing adversarial examples." In: *arXiv preprint arXiv:1412.6572* (2014).

[Lev73] L. A. Levin. "Universal sequential search problems." In: *Problemy peredachi informatsii* 9.3 (1973), pp. 115–116.

[Li+22] M. Li, Z. Shi, Q. Lai, S. Khan, and Q. Xu. "DeepSAT: An EDA-Driven Learning Framework for SAT." In: *arXiv preprint arXiv:2205.13745* (2022).

[Mąd+17] A. Mądry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. "Towards deep learning models resistant to adversarial attacks." In: *stat* 1050 (2017), p. 9.

[MMZ06] S. Mertens, M. Mézard, and R. Zecchina. "Threshold values of random K-SAT from the cavity method." In: *Random Structures & Algorithms* 28.3 (2006), pp. 340–373.

[MS99] J. P. Marques-Silva and K. A. Sakallah. "GRASP: A search algorithm for propositional satisfiability." In: *IEEE Transactions on Computers* 48.5 (1999), pp. 506–521.

[PS98] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[SB19] D. Selsam and N. Bjørner. "Guiding high-performance SAT solvers with unsat-core predictions." In: *Theory and Applications of Satisfiability Testing–SAT 2019: 22nd International Conference, SAT 2019, Lisbon, Portugal, July 9–12, 2019, Proceedings 22*. Springer. 2019, pp. 336–353.

[Sel+18] D. Selsam, M. Lamm, B. Bünz, P. Liang, L. de Moura, and D. L. Dill. "Learning a SAT solver from single-bit supervision." In: *arXiv preprint arXiv:1802.03685* (2018).

[Shi+22] Z. Shi, M. Li, S. Khan, H.-L. Zhen, M. Yuan, and Q. Xu. "SATformer: Transformers for SAT Solving." In: *arXiv preprint arXiv:2209.00953* (2022).

[Sze+13] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. "Intriguing properties of neural networks." In: *arXiv preprint arXiv:1312.6199* (2013).

[Wan+21] W. Wang, Y. Hu, M. Tiwari, S. Khurshid, K. McMillan, and R. Miikkulainen. "NeuroComb: Improving SAT Solving with Graph Neural Networks." In: *arXiv preprint arXiv:2110.14053* (2021).

[YGS20] G. Yehuda, M. Gabel, and A. Schuster. "It's Not What Machines Can Learn, It's What We Cannot Teach." In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 10831–10841.