Data Analytics and Machine Learning Group
TUM School of Computation, Information and Technology
Technical University of Munich

Adversarial Training for Neural Combinatorial Solvers

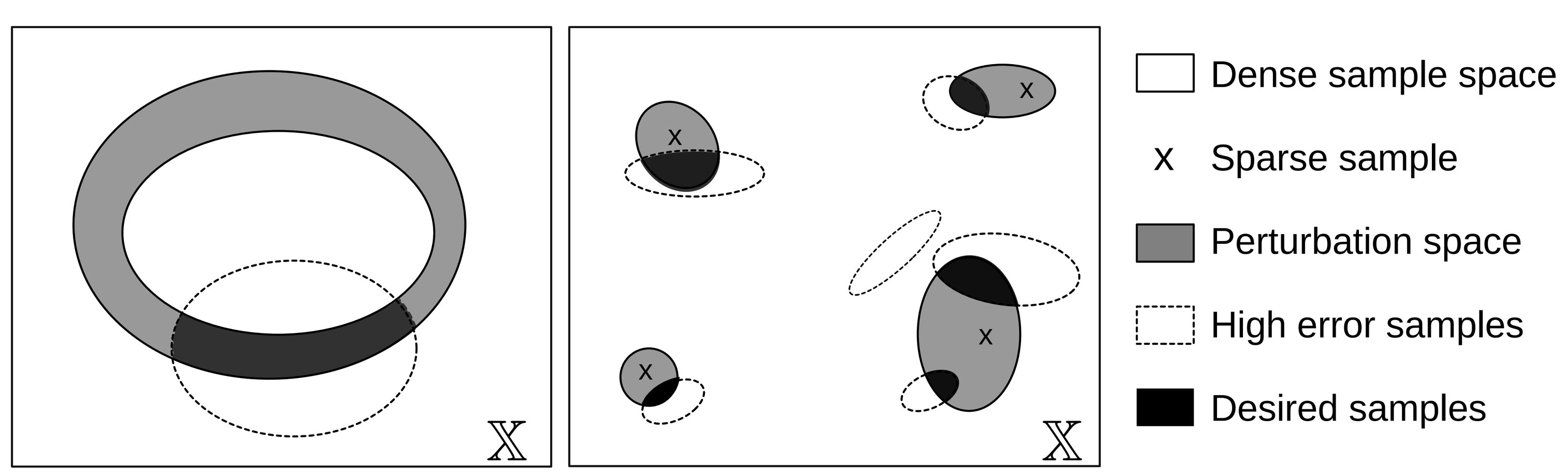
Oriol Aranda Llorens, Johanna Sommer, Stephan Günnemann

TL;DR

- We propose **adversarial training** to improve **generalization** and **robustness** on neural combinatorial solvers.
- We propose a **more efficient** data generation method for SAT and k-SAT.
- We show adversarial training **does not negatively impact** training performance.

Neural Combinatorial Solvers

- **Bad generalization:** from randomly generated data to other domains, and from small to large problem instances.
- **Bad robustness:** adversarial attacks reveal hard model-specific instances
- **Data generation** is either:
  ✗ Incomplete and efficient
  ✗ Complete and inefficient

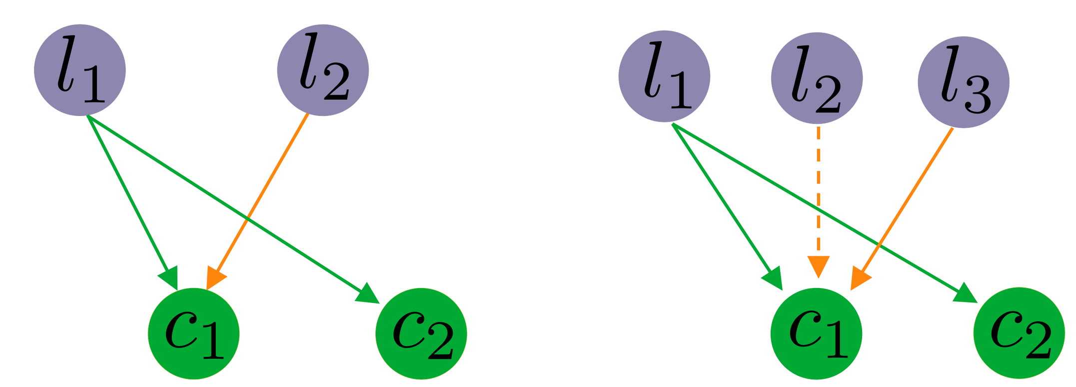


Use adversarial attacks to:
1) improve robustness and generalization
2) ease learning with costly generated data

Adversarial Training

- Exploit SAT **invariances** (satisfiability):

$$(l_1 \lor l_2) \land (l_1) = True \qquad \begin{cases} del \rightarrow (l_1 \lor \cancel{l_2}) \land (l_1) = True \\ add \rightarrow (l_1 \lor l_2 \lor l_3) \land (l_1) = True \end{cases}$$



- Given an instance x and a neural solver S_θ, we obtain the **perturbed instance** $\widetilde{x} = f_p(x)$ learning a perturbation matrix p s.t.

$$\max_p \mathcal{L}(S_\theta(\widetilde{x}))$$

$$f_p(x) = \begin{cases} x_{ij} - p_{ij}, & \text{if edge } i \rightarrow j \text{ exists} \\ x_{ij} + p_{ij}, & \text{otherwise} \end{cases}$$

$$x \left\{ \begin{array}{r|cc|} & c_1 & c_2 \\ l_1 & 1 & 1 \\ l_2 & 1 & 0 \\ \overline{l_1} = l_3 & 0 & 0 \\ \overline{l_2} = l_4 & 0 & 0 \end{array} \right. \overset{f_p}{\Longrightarrow} \widetilde{x} \left\{ \begin{array}{|cc|} c_1 & c_2 \\ 1-0 & 1-0 \\ 1-p_{21} & 0+p_{22} \\ 0+p_{31} & 0+p_{32} \\ 0+p_{41} & 0+p_{42} \end{array} \right.$$

- Adversarial training procedure: use the perturbed instances to train the neural solver S_θ s.t.

$$\min_\theta \mathcal{L}(S_\theta(\widetilde{x}))$$

Efficient Data Generator

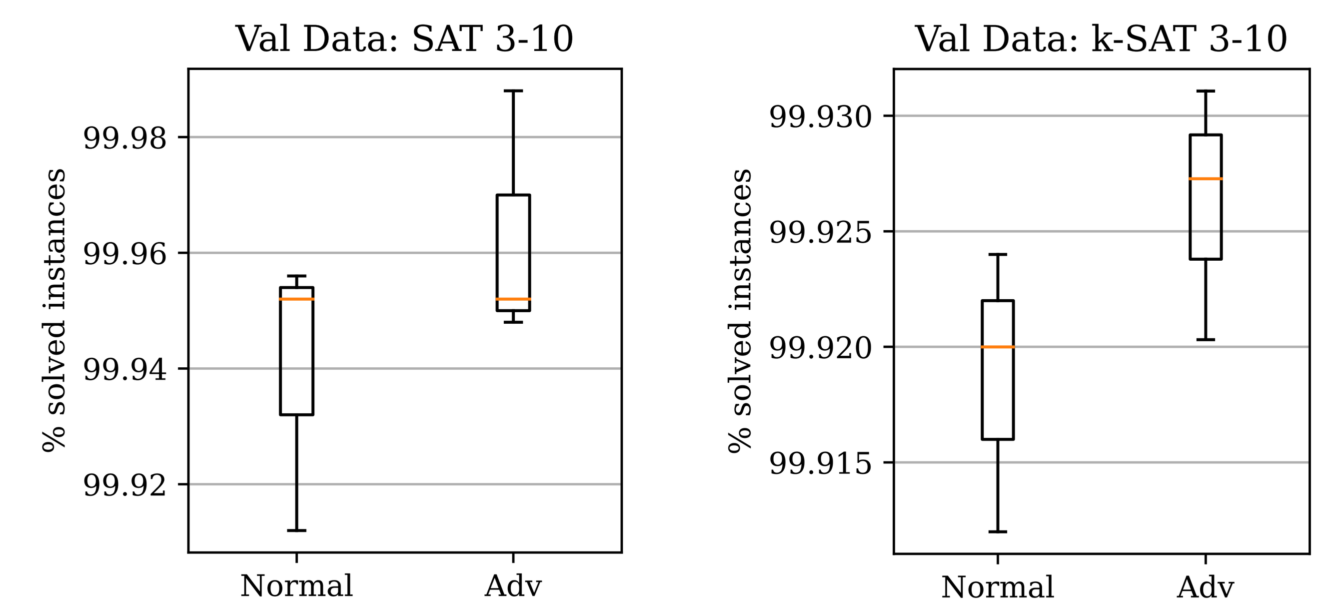
- **Former method:** randomly build x and get the solution y with a classical solver → Inefficient
- **Our method:** randomly build x starting from a solution y → No need for a solver

$$y = [x_1, \overline{x_1}, x_2, \overline{x_2}, x_3, \overline{x_3}]$$

$$x = \bigwedge_{i=1}^{n_c} (l_{i1} \lor l_{i2} \lor \cdots \lor l_{ik_i})$$

- For SAT $\forall i,\ k_i \sim 1 + Ber(0.7) + Geo(0.4)$ and $n_c \sim N(\mu, \sigma^2)$
- For k-SAT $\forall i,\ k_i = k$ and $n_c = n_v \cdot \alpha_k$

Training Performance



- **Adversarial training does not negatively impact** training performance since we have obtained same results compared with the normal training.