

Advances in Neuro Symbolic Reasoning (*and Learning*)



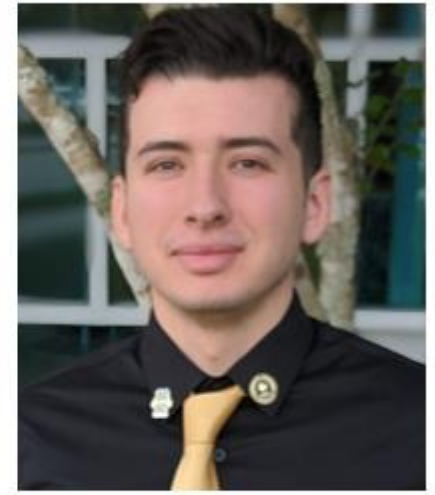
Paulo Shakarian, ASU



Gerardo I. Simari, UNS



Chitta Baral, ASU



Alvaro Velasquez, DARPA

Overview

- 8:30-8:45 Tutorial Overview (Shakarian)
- 8:45-9:30 Introduction and overview of neuro symbolic frameworks for reasoning and learning (Shakarian)
 - LNN, Annotated Logic, dILP, DSP, SATNet
- 9:30-9:40 Break
- 9:40-10:40 Neuro symbolic based approaches for deduction (Simari)
 - LTN and deep ontological networks
- 10:40-10:50 Break
- 10:50-11:50 Combining perceptual neural networks with logic and applications (Baral)
 - NeurASP, and NLP/VQA applications
- 11:50-12:00 Break
- 12:00-12:30 Neuro Symbolic Reasoning for the DoD (DARPA) (Velasquez)

Resources: Tutorial Web Page

- <https://labs.engineering.asu.edu/labv2/2023-aaai-tutorial-advances-in-neuro-symbolic-reasoning/>

Lab V2

Home

About Paulo Shakarian

Selected Readings

Before Lab V2

Interested Students

Neuro Symbolic Reasoning Resource Page

Lab Members

2023 AAAI Tutorial: Advances in Neuro Symbolic Reasoning

Machine Learning Fairness Resources

2023 AAAI Tutorial: Advances in Neuro Symbolic Reasoning

[Home](#) » 2023 AAAI Tutorial: Advances in Neuro Symbolic Reasoning

We are proud to present a 2023 AAAI Tutorial [TFHA6: Advances in Neuro Symbolic Reasoning](#)

- Date: **Tuesday, Feb. 7, 2023**
- Time: **8:30am-12:30pm (half-day)**
- Location: **Walter E. Washington Convention Center, Washington DC, USA**
- YouTube (post-tutorial): <https://www.youtube.com/@neurosymbolic>

This resource page will be updated periodically prior to the event.

Tutorial Description

Over the past five years, the community has made significant advances in neuro symbolic reasoning (NSR). These NSR frameworks are now capable of embedding prior knowledge in deep learning architectures, guiding the learning process with logical constraints, providing symbolic explainability, and using gradient-based approaches to learn logical statements. At this time, several approaches are seeing usage in various application areas. This tutorial is designed for researchers looking to understand the current landscape of NSR research as well as those looking to apply NSR research in areas such as natural language processing and verification. The pace of progress in NSR is expected to continue as firms such as IBM, Samsung, and Lockheed-Martin are now heavily investing in research in this area in addition to the recently announced government-funded efforts such as DARPA's ASNR program indicate that this area will grow. A secondary goal of this tutorial is to help build a larger community around this topic as more basic researchers and applied scientists turn to NSR to build upon the successes of deep learning. Attendees of the tutorial should be familiar with concepts in deep learning and logical reasoning, have mathematical maturity, as well as a basic understanding of fuzzy/real-valued logic.

Resources: YouTube Channel

- <https://www.youtube.com/@neurosymbolic>

NEURO SYMBOLIC

Neuro Symbolic
@neurosymbolic
193 subscribers

Subscribed

HOME VIDEOS PLAYLISTS COMMUNITY CHANNELS ABOUT

Videos ▶ Play all

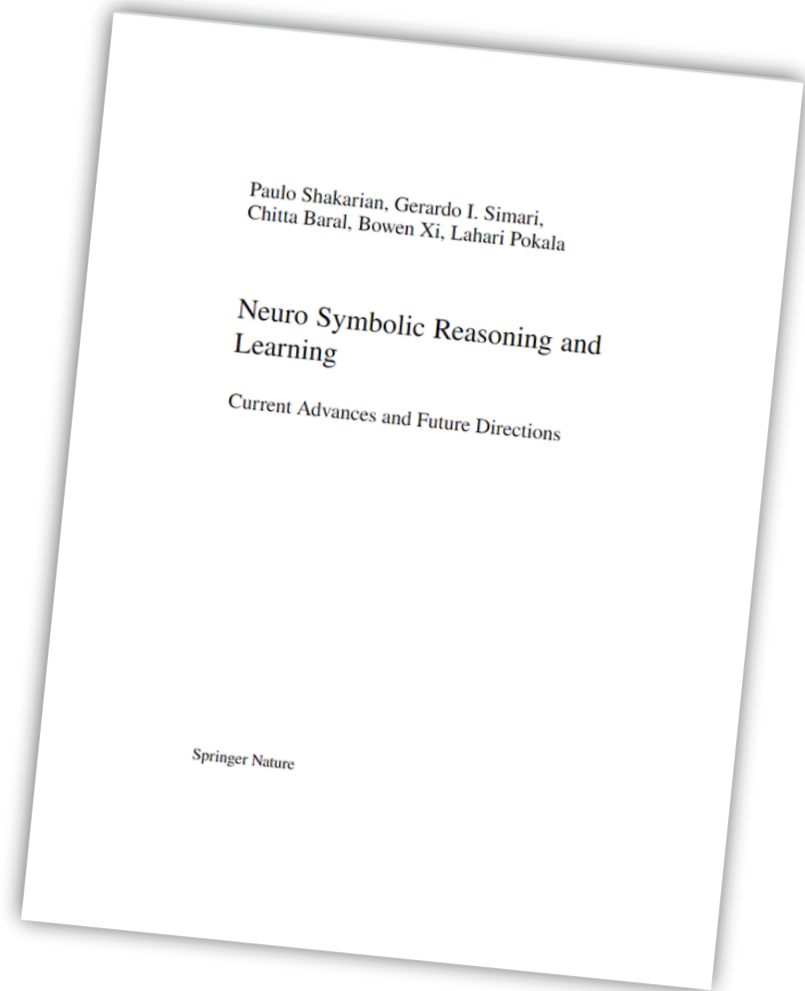
- Differentiable ILP Pt 3: Neural Architecture**
10 views • 14 hours ago
- Luis Lamb Part 11: Neuro symbolic AI vs...**
43 views • 2 days ago
- Luis Lamb Part 10: Can Neuro Symbolic approaches...**
47 views • 7 days ago
- Differentiable ILP Pt 2: ILP by SAT Solving**
19 views • 9 days ago
- Differentiable ILP Pt 1: Introduction**
46 views • 2 weeks ago
- Five overlooked AI papers from 2022**
121 views • 2 weeks ago

Neuro Symbolic Frameworks

- PSL/NeuPSL**
Neuro Symbolic
View full playlist
- SAT Net**
Neuro Symbolic
View full playlist
- Logical Neural Networks**
Neuro Symbolic
View full playlist
- Logic Tensor Networks**
Neuro Symbolic
View full playlist
- NeurASP**
Neuro Symbolic
View full playlist
- NSR for Ontological Reasoning**
Neuro Symbolic
View full playlist

Resources: Book

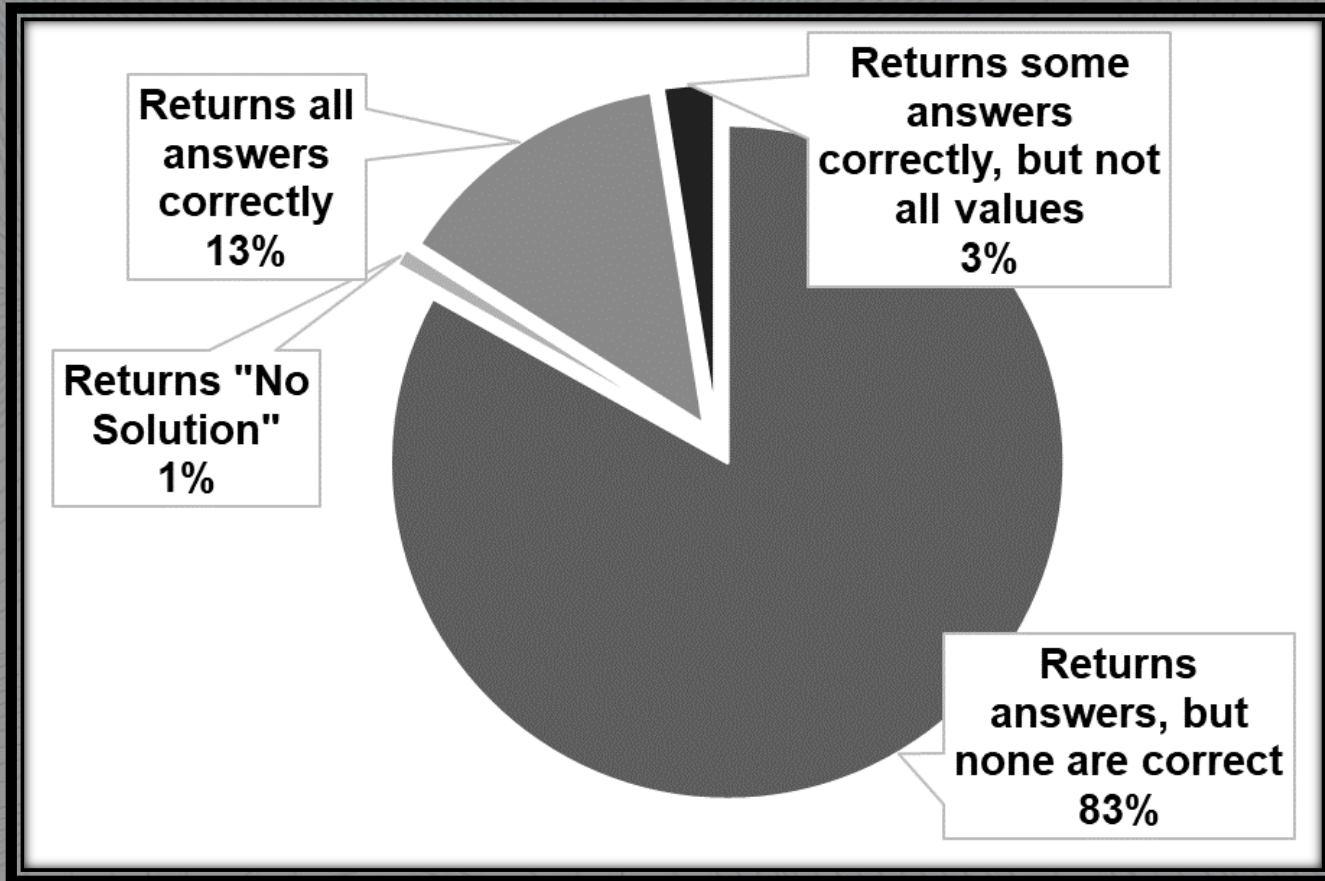
- Anticipated release in late Spring/early Summer 2023
- Published by Springer-Nature



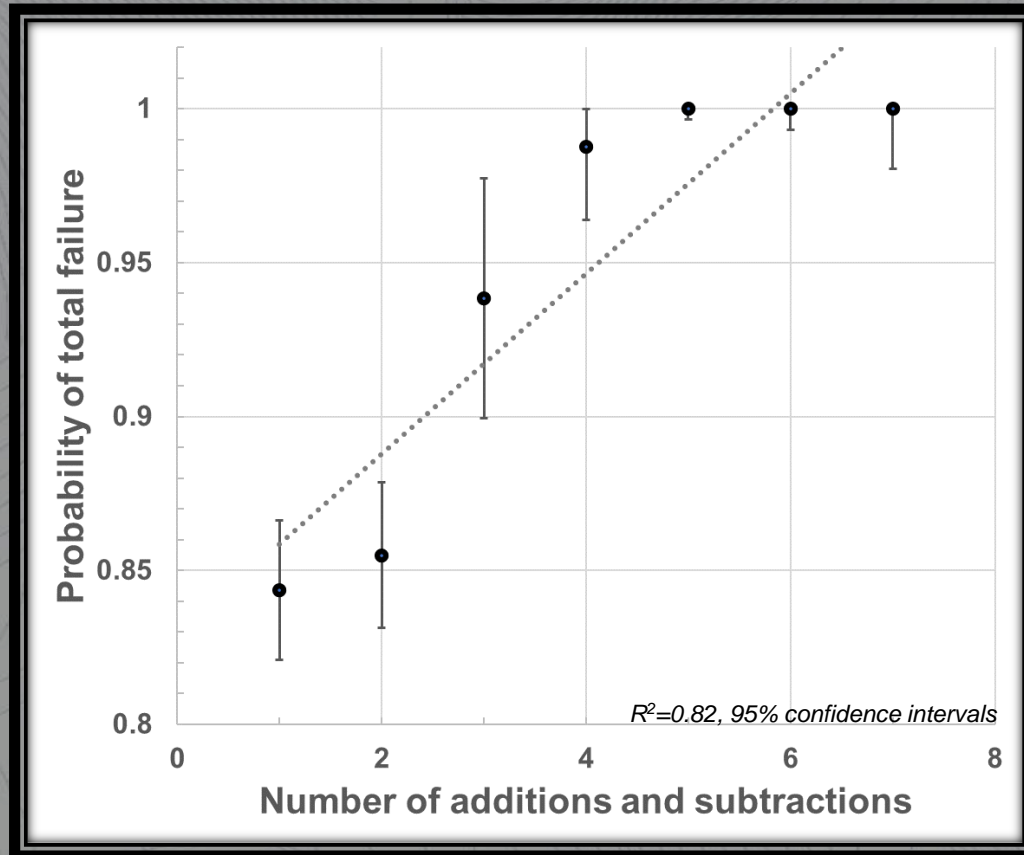
Why Neuro Symbolic?

- Neural Methods provide great results, but often have difficulties in reasoning, explainability, and modularity.
- Symbolic methods excel in the above areas, but have difficulty coping with noise and deriving robust models from real-world data
- Can we have the best of both worlds?

ChatGPT Failures on Math Word Problems



More Additions → More Failures for ChatGPT



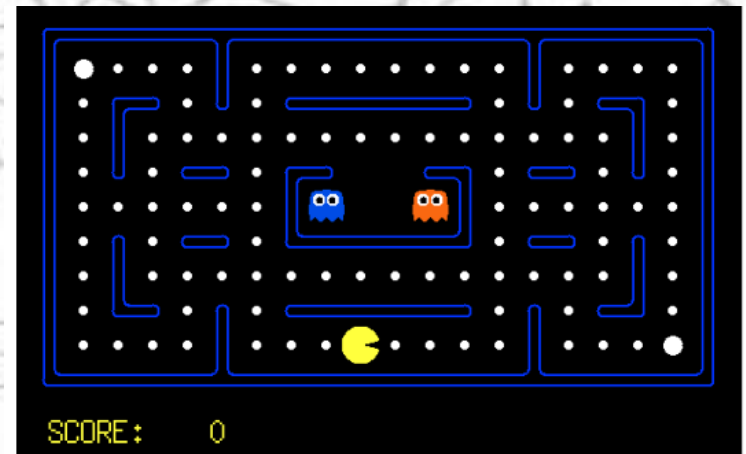
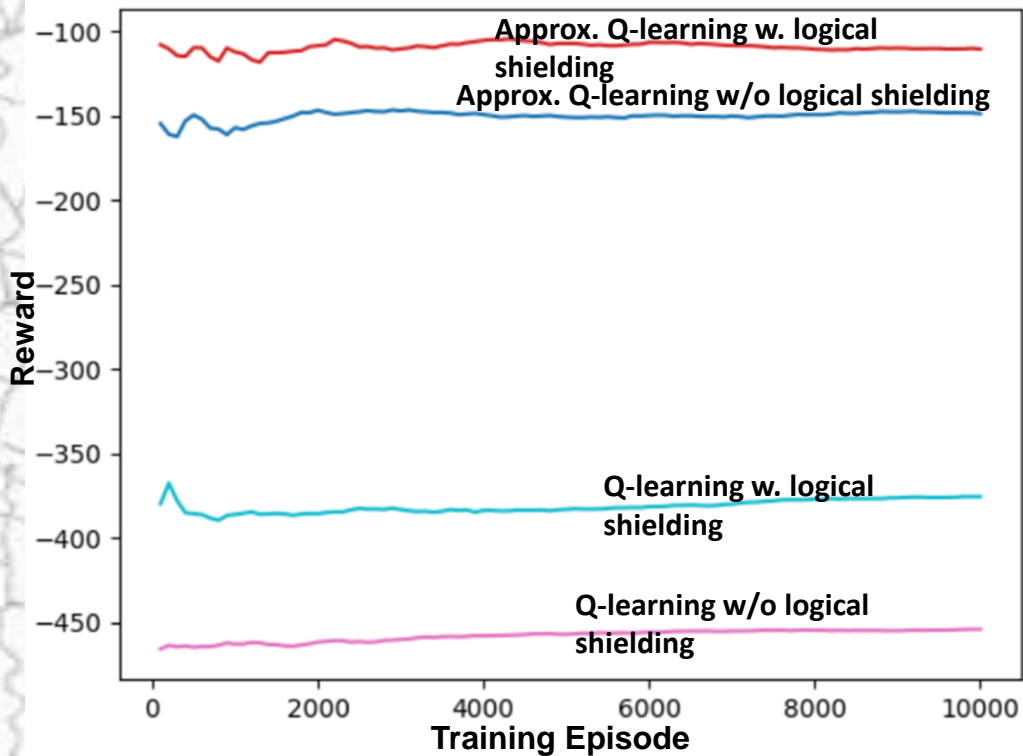


Can we tell Pac-Man to not take dumb moves?

**Idea: express such restrictions in logic.
→ This is called “Logical shielding”**

Will it make the Pac-Man playing AI work better?

Pac Man Performance on “Medium” Difficulty Board



Thanks to Tanmay Khandait, Devendra Rajendra Parkar, and Kirby Kuznia for allowing us to share the results of their student project from CSE 591.

Introduction and overview of neuro symbolic frameworks for reasoning and learning

Paulo Shakarian

Associate Professor

Arizona State University

pshak02@asu.edu

AAAI 2023, Tutorial Section

Overview

- Generalizing differentiable logic with annotated logic
- Logical Neural Networks
- Differentiable Inductive Logic Programming
- Deep Symbolic Policy Learning
- Learning with STL Constraints: STL Net
- Learning Constraints to Combinatorial Problems: SAT Net

Generalizing Differentiable Logic with Annotated Logic

Papers:

M. Kifer, V.S. Subrahmanian, Theory of Generalized Annotated Logic Programs and its Applications. Journal of Logic Programming, Elsevier, 1992.

P. Shakarian, G. Simari, Extensions to Generalized Annotated Logic and an Equivalent Neural Architecture, IEEE TransAI, 2022.

D. Aditya, K. Mukherji, S. Balasubramanian, A. Chaudhary, P. Shakarian, PyReason: Software for Open World Temporal Logic, AAAI Spring Symposium (Mar. 2023).

Annotated Logic

Logical atoms are “annotated” with values from a lattice structure or functions (“annotation functions”) over such a structure, below is an example of a rule in general annotated logic.

$$pred'_i(X) : [relu(l_1 + l_2 + l_3 - 2), 1] \leftarrow pred_{2i}(X) : [l_1, u_1] \wedge pred_{2i-1}(Y) : [l_2, u_2] \wedge pred_i(X, Y) : [l_3, u_3]$$

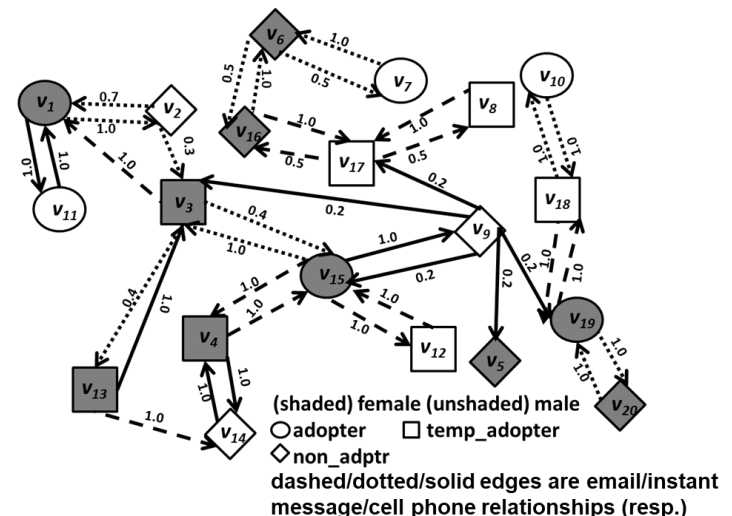
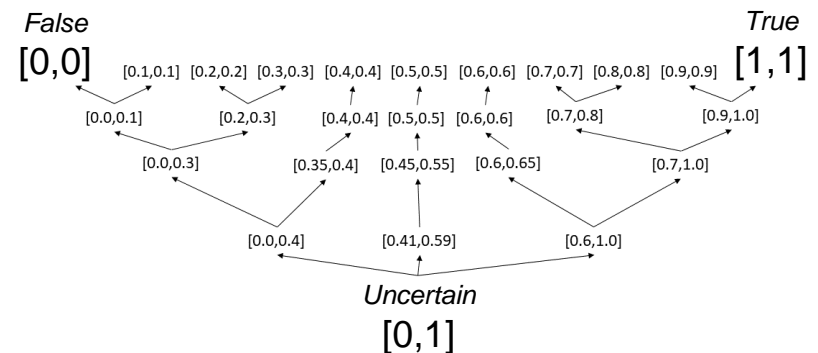
If we specify our lattice as scalars in $[0,1]$ and use T-norms, T-conforms and other fuzzy logic operators as annotation functions, then we can capture various other real-valued logics (e.g., the logics used in LTN’s).

However, provided that we keep the annotation functions differentiable, there framework offers more flexibility.

Kifer and Subrahmanian 1992 show results in the general case (annotations from arbitrary lattice structures), specifically showing that a fixpoint operator provides exact deductive inference.

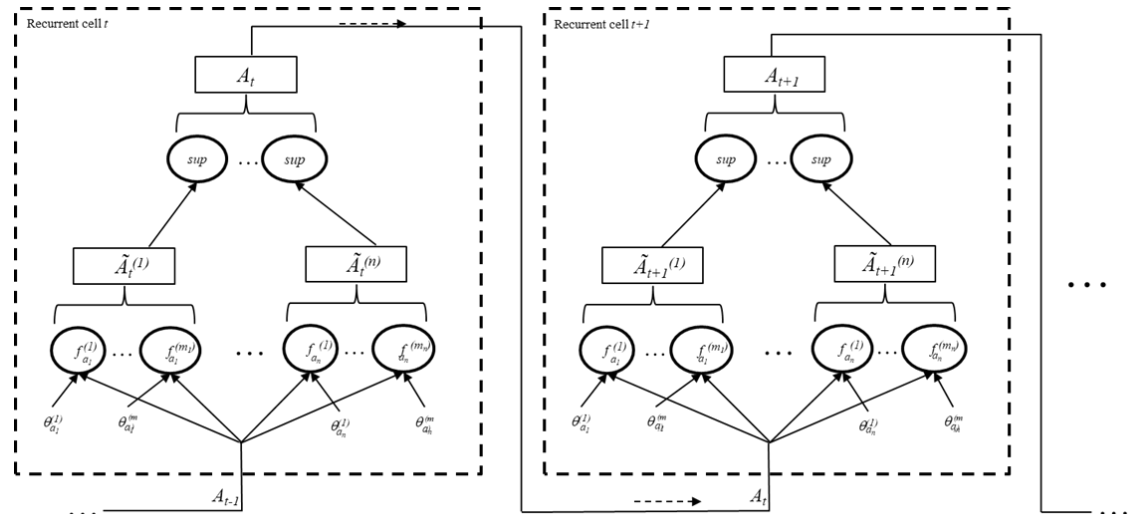
Annotated Logic Enables Open-World Reasoning

- Several papers have associated logical atoms with subsets of the $[0,1]$ interval
- MANCaLog (Shakarian et al., 2013) proposed that with annotated logic for knowledge graph reasoning
- LNN's also use intervals associated with logical atoms
- A key advantage over scalars is that this approach permits open-world novelty



Neural Equivalent Architecture

- Building on ideas from differentiable ILP, we proposed an architecture to leverage annotated logic for rule learning
- Key idea: a recurrent neural unit aligns with a deductive fixpoint operator





PyReason: Python-based temporal first-order logic explainable AI system supporting uncertainty, open-world novelty, and graph-based reasoning.

- Supports generalized annotated logic with temporal, graphical and uncertainty extensions, capturing a wide variety of fuzzy, real-valued, interval, and temporal logics
- Modern Python-based system supporting reasoning on graph-based data structures (e.g., exported from Neo4j, GraphML, etc.)
- Rule-based reasoning in a manner that support uncertainty, open-world novelty, non-ground rules, quantification, etc., agnostic to selection of t-norm, etc.
- Fast, highly optimized, correct fixpoint-based deduction allows for explainable AI reasoning, scales to graphs with over 30 million edges

GitHub:

<https://github.com/lab-v2/pyreason>

Python:

```
pip install pyreason
```

PyReason

Supports Generalized Annotated Logic

Finite-time temporal logic (e.g., as used in STLNet)

Supports inference with fuzzy operators (e.g., LTN))

Supports open world novelty and parameterized operators (e.g., LNN)

Supports graph-based reasoning (e.g., MANCALog)

Logical Neural Networks

Papers:

Riegel, R., Gray, A., Luus, F., Khan, N., Makondo, N., Akhalwaya, I.Y., Qian, H., Fagin, R., Barahona, F., Sharma, U., Ikbal, S., Karanam, H., Neelam, S., Likhyani, A., Srivastava, S.: Logical neural networks (2020).

Sen, P., Carvalho, B.W.S.R.d., Rabdelaziz, I., Kapanipathi, P., Roukjos, S., Gray, A.: Logical Neural Networks for Knowledge Base Completion with Embeddings and Rules, EMNLP (2022).

LNN's

Key ideas:

- Each input to an operator is associated with a parameter (“importance weighting”)
- Forward pass is a deduction algorithm (equivalent to the fixpoint operator of general annotated programs)
- Hyperparameter α sets a threshold for truth and falsehood
- Logic program known a priori (like in LTN's)
- Learning process finds parameters such that operators retain classical functionality

a	b	$a \vee b$
0	0	0
0	1	1
1	0	1
1	1	1

Classical functionality for fuzzy disjunction

a	b	$a \vee b$
$[0, 1-\alpha]$	$[0, 1-\alpha]$	$[0, 1-\alpha]$
$[0, \alpha)$	$[\alpha, 1]$	$[\alpha, 1]$
$[\alpha, 1]$	$[0, \alpha)$	$[\alpha, 1]$
$[\alpha, 1]$	$[\alpha, 1]$	$[\alpha, 1]$

Classical functionality for LNN disjunction

Parameterized Lukasiewicz-style Fuzzy Operators used in LNN's with a generic activation function (f)

Strong Negation (used in LNN without parameters):

$$N(x) = 1 - x \quad (6.1)$$

LNN variant of the Lukasiewicz t-norm w. activation function:

$$T_{LNN}(\{x_1, \dots, x_n\}) = f\left(\beta + \sum_i w_i (x_i - 1)\right) \quad (6.2)$$

LNN variant of the Lukasiewicz t-conorm w. activation function:

$$S_{LNN}(\{x_1, \dots, x_n\}) = f\left(1 - \beta + \sum_i (w_i \cdot x_i)\right) \quad (6.3)$$

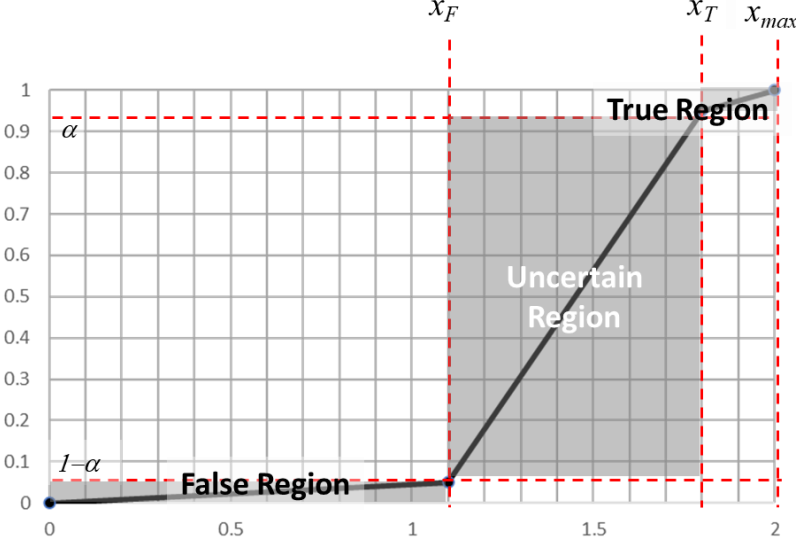
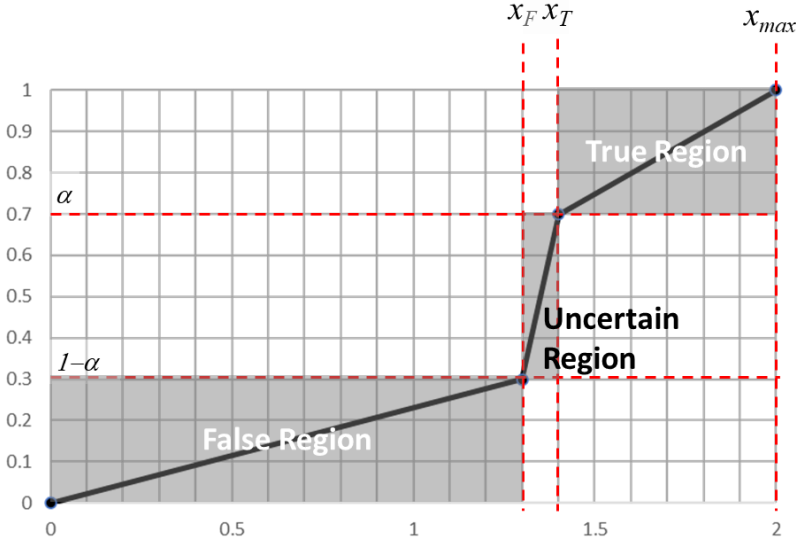
LNN variant the Lukasiewicz implication w. activation function:

$$I_{LNN}(x_1, x_2) = f(1 - \beta + w_1(1 - x_1) + w_2 x_2) \quad (6.4)$$

LNN: Inference

- Input to inference:
 - Set of formulas
 - Initial truth bounds for each atom and formula
 - Bias and weight for each formula (connector)
- Output:
 - Final truth bounds for each formula and atom
- Authors propose a upward-downward pass through the logic (*this is different from forward-backward pass used in gradient descent*)
- The algorithm propagates truth values from atoms to the formula (upward) and from the formulas to the atoms (downward) until convergence

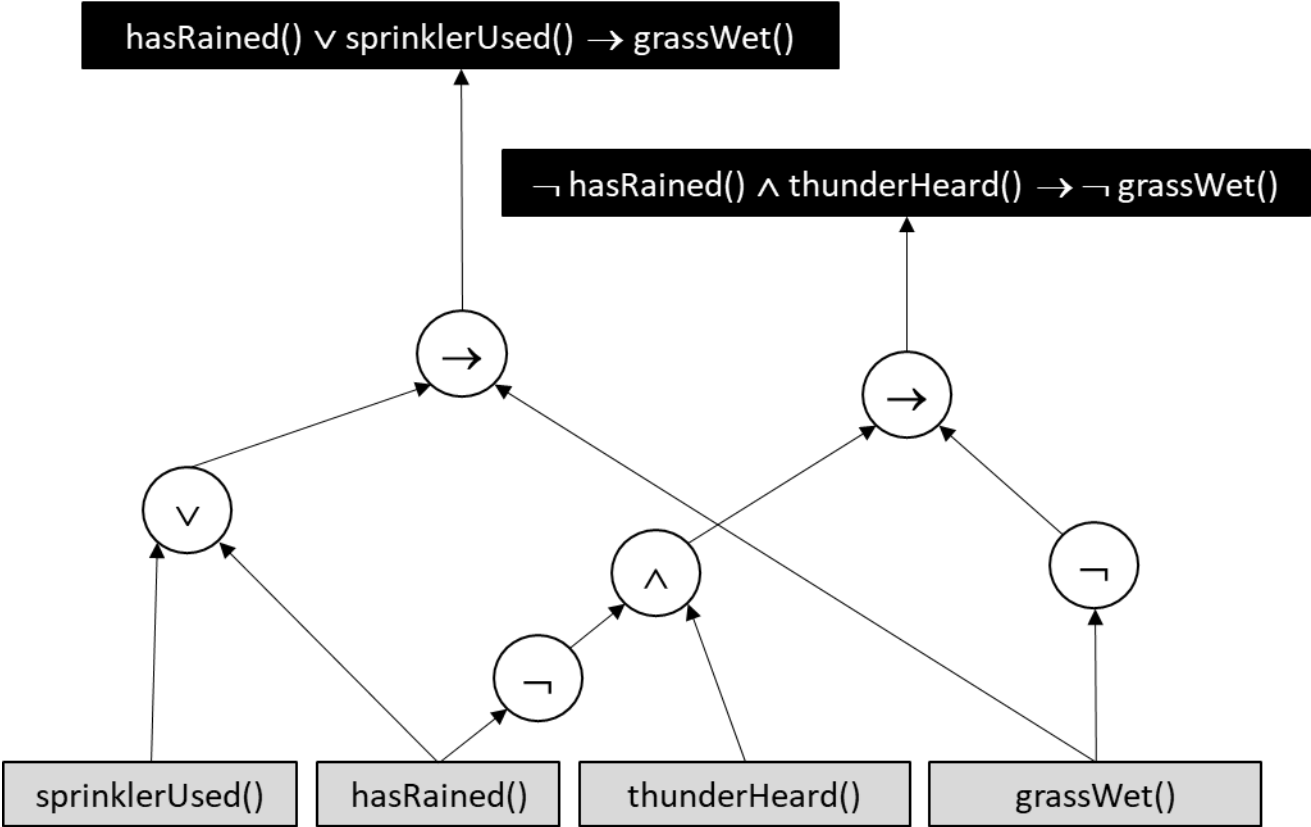
LNN: Activation Functions



LNN: Learning

- Neural architecture is derived directly from the a-priori known formulas
- Historical inputs and outputs used as samples during the training process
- Loss function depends not only on standard ML metrics (e.g., MSE) but also the number of inconsistencies (neurons associated with a truth bound where $L > U$).
- The functions used to combine formulas are differentiable with respect to the weights
- The forward pass of the learning process can be done with the inference algorithm (which is essentially a fixpoint operator)

LNN: Syntax Tree / Neural Structure



Objective Function with Constraints

- To ensure reasonable settings of parameters, the authors show how the parameter-learning problem can be framed as an optimization problem with constraints

$$\begin{aligned} \min_{B,W} \quad & E(B, W) + \sum_{k \in N} \max\{0, L_{B,W,k} - U_{B,W,k}\} \\ \text{s.t.} \quad & \forall k \in N, i \in I_k, \quad \alpha \cdot w_{ik} - \beta_k + 1 \geq \alpha, \quad w_{ik} \geq 0 \end{aligned} \quad (6)$$

$$\forall k \in N, \quad \sum_{i \in I_k} (1 - \alpha) \cdot w_{ik} - \beta_k + 1 \leq 1 - \alpha, \quad \beta_k \geq 0 \quad (7)$$

- Here, $E(B, W)$ is the traditional loss function and the summation $\sum_{k \in N} \max\{0, L_{B,W,k} - U_{B,W,k}\}$ is designed to reduce the number of inconsistencies
- Some Issues: (fully listed in section F.1)
 - Requirement of additional slack parameters
 - Parameter updates require constraint satisfaction
 - β must be learnt for each neuron, hinders interpretability and leads to overfitting

Tailored Activation Function

Shown here is a tailored activation function for disjunction with $\beta=1$.

$$f_{\mathbf{w}}(x) = \begin{cases} x \cdot (1 - \alpha) / x_{\text{F}} & \text{if } 0 \leq x \leq x_{\text{F}}, \\ (x - x_{\text{F}}) \cdot (2\alpha - 1) / (x_{\text{T}} - x_{\text{F}}) + 1 - \alpha & \text{if } x_{\text{F}} < x < x_{\text{T}}, \\ (x - x_{\text{T}}) \cdot (1 - \alpha) / (x_{\text{max}} - x_{\text{T}}) + \alpha & \text{if } x_{\text{T}} \leq x \leq x_{\text{max}}, \end{cases}$$

$$x_{\text{F}} = \sum_{i \in I} w_i \cdot (1 - \alpha), \quad x_{\text{T}} = w_{\text{max}} \cdot \alpha, \quad x_{\text{max}} = \sum_{i \in I} w_i.$$

Key advantages:

- The logic is maintained regardless of the weights
- The authors claim that the output is independent of β , so they define it

as $\beta_C = \sum_{i \in I} w_i$

Overall Strategy

- Use gradient descent to find weights, using normal back-propagation and the aforementioned inference process for the forward pass
- Loss function combines normal metrics (e.g., MSE) and a count of inconsistent neurons

Key Issues to Consider

- Parameters w and β must be set in a way such that classical logic outcomes for the operators behave as expected
- Learning parameters that fit vs. interpretability

Inconsistency

- Not fundamentally guaranteed
- Authors do mention that consistency check can be done during the training process
- It is noteworthy that consistency is based on neurons, not atoms – so for all formulas known a-prior, you know which ones will be inconsistent
- However, if you are checking an entailment query against the logic, there are no guarantees if it is correct

Recent Work

- Sen et al. (2022) extend LNN's for rule-learning in knowledge graphs and achieve state-of-the-art performance on KBC tasks.
- Approach is complementary to embedding-based approaches to the problem – and the combination provides further improvement.

Differentiable Inductive Logic Programming

Paper:

Evans, R., Grefenstette, E.: Learning explanatory rules from noisy data. *J. Artif. Int. Res.* 61(1), 1–64 (2018)

The Logic of δ ILP

First-Order Logic (predicates and constants)

- Given predicates p, q constant c and variable x
- $p(c)$ is a ground atom
- $p(x)$ is non-ground
- $q(x) \leftarrow p(x)$ is a non-ground rule (“if p then q”)
 - Each non-ground rule will have multiple grounded instances
- Assume facts (ground atoms) and (non-ground) rules
- Limits on the number of inference steps (i.e., applications of a fixpoint operator) – denoted T

ILP by SAT Solving

Intuition:

1. A “template” specifies the format of a rule (e.g., number of body predicates, free existentially quantified variables, number of invented predicates, etc.)
2. For a given predicate, “candidate” rules are generated based on a template

$$r_i \equiv p(X) \leftarrow q_1(X) \wedge \dots \wedge q_m(X)$$

3. An additional atomic proposition is added to the end of each candidate

$$r_i \equiv p(X) \leftarrow q_1(X) \wedge \dots \wedge q_m(X) \wedge f_i$$

4. Given positive and negative facts, find a subset of the propositions that “turn on” candidate rules such that positive facts are entailed and negative facts are not

$$\forall a \in \mathcal{P} : \mathcal{B} \cup F' \cup \Pi'' \models a$$

$$\forall a \in \mathcal{N} : \mathcal{B} \cup F' \cup \Pi'' \not\models a$$

Learning Problem

- The overall goal is to compute the following conditional probability

$$p(\lambda | a, W, \Pi, \mathcal{B})$$

Truth value for
ground atom a

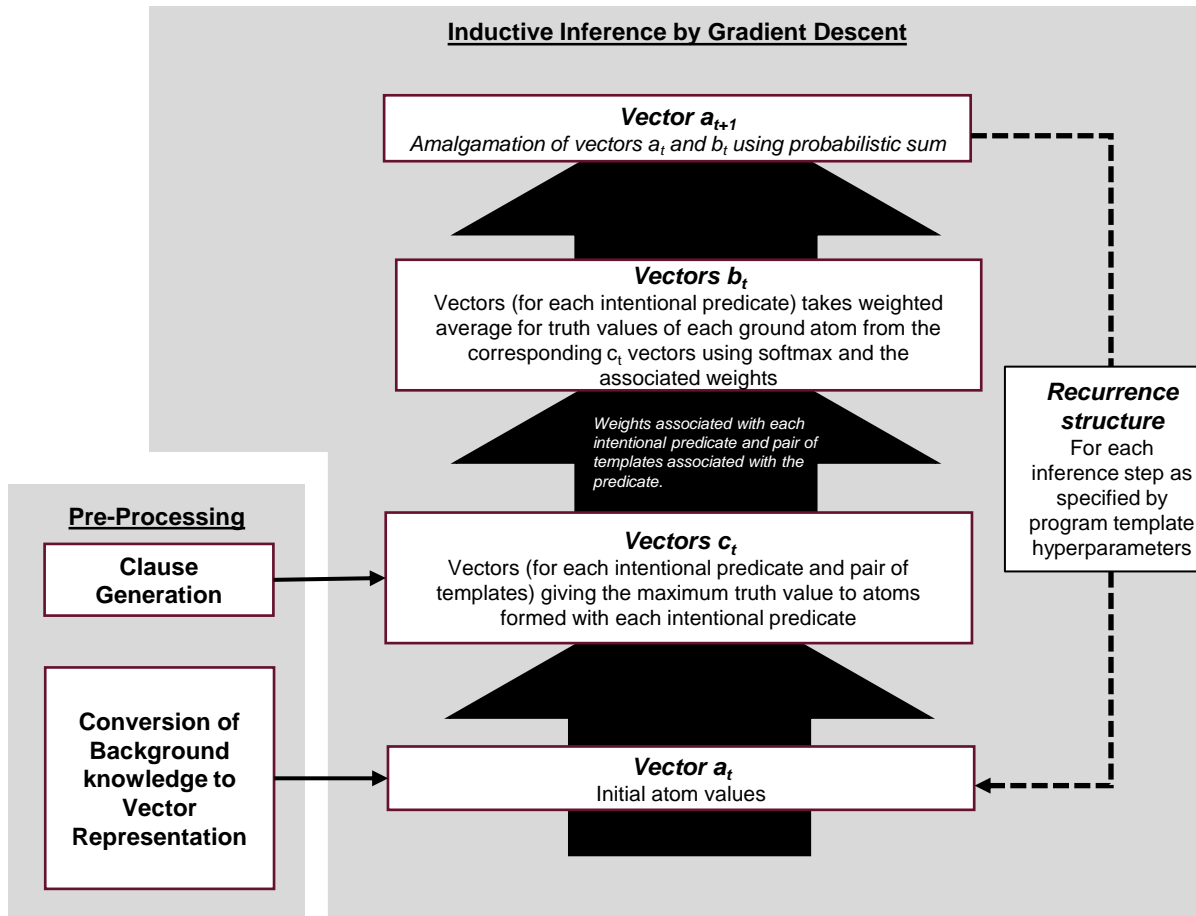
Weights (learned), program
template, language, and facts

- Such that the following loss is minimized

$$-\mathbb{E}_{(a, \lambda) \sim \Lambda} [\lambda \cdot \log(p(\lambda | a, W, \Pi, \mathcal{B})) + (1 - \lambda) \cdot \log(1 - p(\lambda | a, W, \Pi, \mathcal{B}))]$$

Atom, value
(in $\{0,1\}$)
pairs in
ground truth

Overall Architecture



Bounding the Number of Templates

- A major source of complexity is the number of templates, which can be bounded by various quantities as follows:

$$\begin{aligned} \sum_{i=1}^{|P_{in}|} \prod_j^{num_\tau} |cl(\tau_i^j)| &\leq \sum_{i=1}^{|P_{in}|} \prod_j^{num_\tau} ((|P_{ext}| + int_i)(arity_i + v)^{arity_{max}})^{body_{max}} \\ &\leq |P_{in}| \cdot (|P|(arity_{max} + v)^{arity_{max}})^{body_{max} \cdot num_\tau} \end{aligned}$$

- Under the assumptions in the paper, the above bound is equal to the following:

$$6561 \cdot |P_{in}| \cdot |P|^4$$

Experimental Notes

- Training data:
 - Training data consists of multiple $(\mathcal{B}, \mathcal{P}, \mathcal{N})$ triples
 - At each step, one of the triples is sampled
 - From the sampled triple, a mini batch of \mathcal{PUN} is selected
 - Authors state that this method helps escape local minima
 - Training occurs in 6,000 steps
- Other notes:
 - Cross-entropy loss (seen in other work as well)
 - RMS Prop used as optimizer with a learning rate of 0.5
 - Adam also gave reasonable results
 - RMS Prop performed well with lower learning rates (e.g., 0.01)
 - Clause weights initialized from a normal distribution over the interval $[0, 1]$ (mean zero, sd between 0 and 2)

Deep Symbolic Policy Learning

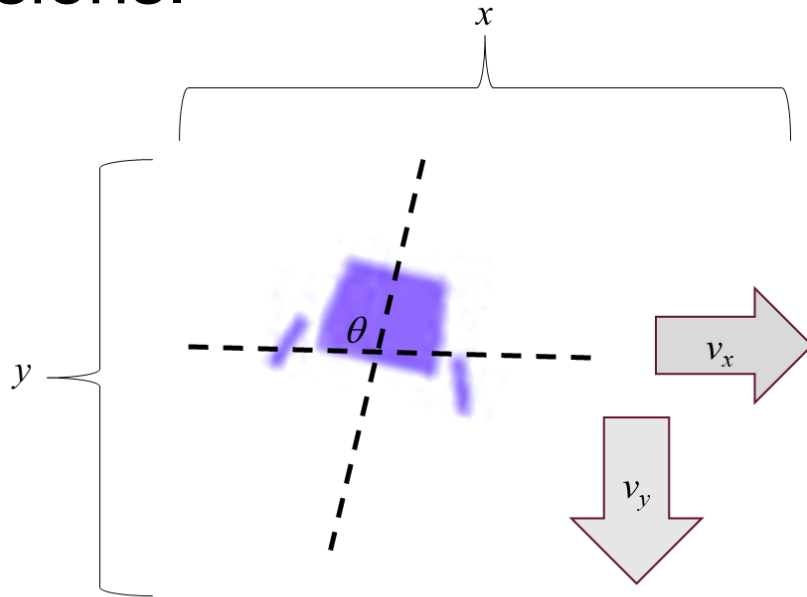
Papers:

Petersen, B.K., Larma, M.L., Mundhenk, T.N., Santiago, C.P., Kim, S.K., Kim, J.T.: Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. arXiv preprint arXiv:1912.04871 (2019)

Landajuela, M., Petersen, B.K., Kim, S., Santiago, C.P., Glatt, R., Mundhenk, N., Pettit, J.F., Faissol, D.: Discovering symbolic policies with deep reinforcement learning. In: International Conference on Machine Learning, pp. 5979–5989. PMLR (2021)

DSP: Motivating Example

Key idea: Understanding physical systems based on data with compact mathematical expressions.



$$a_1 = -10s_2 + \sin(s_3) - 14s_4 - 1.99$$
$$a_2 = -5.79 \frac{s_4}{s_6 - s_3}$$

State	Definition
s_1	x
s_2	y
s_3	v_x
s_4	v_y
s_5	θ
s_6	$\frac{d\theta}{dt}$
s_7	First leg contact
s_8	Second leg contact

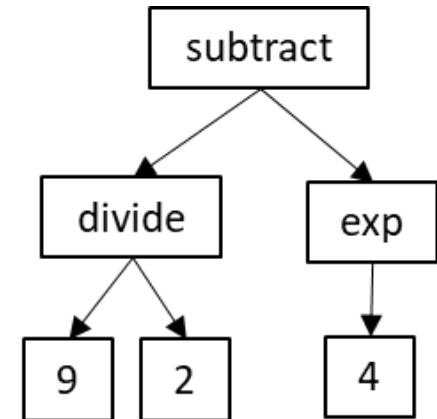
Action	Definition
a_1	Main engine
a_2	Engine left or right

Expression Tree

An expression tree is a syntax tree for mathematical expressions.

Unlike LNN's where the syntax tree is given and embedded into the NN, in DSO/DSR we learn the tree using an RNN in a RL framework.

$$\frac{9}{2} - e^4$$

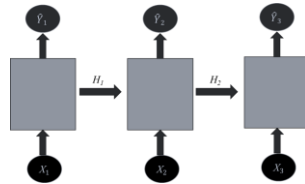


Why Symbolic Policies for Control?

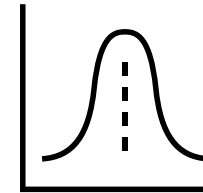
- Traditional control theory and mathematical physics approaches for control result in simple but effective models
- Further, these models are mathematical equations that are simple (hence regularized), easily understood, and can be efficient to implement
- Prior work on RL for control results in black-box models that do not have these features

Deep Symbolic Regression

Update RNN parameters



Generate distribution of expressions with an RNN



Compute gradient using top epsilon expressions via “risk-seeking” reward function

$$\nabla_{\theta} J_{risk}(\theta; \varepsilon)$$

Evaluate reward associated with expressions based on NRMSE to identify top epsilon expressions

$$\frac{1}{\sigma} \sqrt{\frac{1}{n} \sum_i (y_i - f(X_i))^2}$$

Reward for Individual Expressions

$$R(\tau) = \frac{1}{N_{ep}} \sum_{i=1}^{N_{ep}} \sum_{t=1}^{lth(i)} r_t(i)$$

Number of episodes

Time steps
in episode i

Reward for time t
in episode i

“Risk-Seeking” Policy Gradient

Standard policy gradient is based on an overall expected value

CVaR policy gradient considers only the lowest risk candidates in a given sample (Tamar et al., 2014)

This work uses a risk-seeking policy gradient that is looking at the highest-reward expressions

Risk-seeking reward function

$$J_{risk}(\theta; \epsilon) = \mathbb{E}_{\tau \sim p(\tau|\theta)} [R(\tau) | R(\tau) \geq R_{\epsilon}(\theta)]$$

Gradient

$$\nabla_{\theta} J_{risk}(\theta; \epsilon) = \mathbb{E}_{\tau \sim p(\tau|\theta)} [(R(\tau) - R_{\epsilon}(\theta)) \cdot \nabla_{\theta} \log p(\tau|\theta) | R(\tau) \geq R_{\epsilon}(\theta)]$$

Adding Multiple Dimensions

- The authors note that multiple action dimensions leads to a combinatorial explosion
- They overcome the problem using a non-symbolic “anchor model”
- The intuition is that each action dimension is learned sequentially.
 - When action dimension i is learned, the algorithm uses previously learned symbolic actions $1, \dots, i-1$ and the anchor (non-symbolic NN-learned) actions for $i+1, \dots, n$.

Deep Symbolic Policies

1. Black box anchor policy learned for all action dimensions

Black box model learned that maximizes *eval*
 $\Psi = \arg \max_{\Psi'} \text{eval}(\Psi')$

*Black box model holds place
of second dimension*



2. Symbolic policy for action dimension 1 is learned

Reward function for single dimension; uses black box model for the second dimension.

$$R(\tau) = \text{eval}(\langle \tau, \Psi_2 \rangle)$$

Pick symbolic expression that maximizes reward

$$f_1 = \arg \max_{\tau} (R(\tau))$$

Selected expression in Lunar Lander example

$$f_1(S) = -10s_2 + \sin(s_3) - 14s_4 - 1.99$$

*Symbolic expression holds
place of first dimension*



3. Symbolic policy for action dimension 2 is learned

Reward function for single dimension; uses learned symbolic expression for the first dimension.

$$R(\tau) = \text{eval}(\langle f_1, \tau \rangle)$$

Pick symbolic expression that maximizes reward

$$f_2 = \arg \max_{\tau} (R(\tau))$$

Selected expression in Lunar Lander example

$$f_2(S) = -5.79 \frac{s_4}{s_6 - s_3}$$

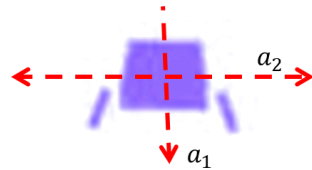
*Returns fully symbolic policy
across all dimensions*



Final Symbolic Policy

$$f_1(S) = -10s_2 + \sin(s_3) - 14s_4 - 1.99$$

$$f_2(S) = -5.79 \frac{s_4}{s_6 - s_3}$$



Notes

$\text{eval}(P)$ Evaluates multi-dimensional action policy P

$\langle P_1, P_2 \rangle$ Each action dimension of policy P

STL Net

Paper:

Ma, M., Gao, J., Feng, L., Stankovic, J.A.: Stlnet: Signal temporal logic enforced multivariate recurrent neural networks. 34th Conference on Neural Information Processing Systems (NeurIPS 2020).

Signal Temporal Logic (STL)

- A temporal logic (like LTL or CTL)
 - Semantic structure assumes multiple worlds or states over time
- Allows for reasoning over time intervals (like MTL)
- Predicates represent *analog signals* meaning that they are equivalent to the value of a function exceeding a certain value (usually zero)

$$\varphi ::= \mu \mid \neg\varphi \mid \varphi' \wedge \varphi'' \mid \varphi' \vee \varphi'' \mid \diamond_{(a,b)}\varphi \mid \square_{(a,b)}\varphi \mid \varphi \mathbf{U}_{(a,b)}\varphi$$

- *Reasonable range.* Values of certain signals should always be within a specified range.
- *Consecutive change.* The change in a value within a certain time period should be within a certain threshold.
- *Temporal Correlation.* Certain correlations known a-priori can be captured by such a constraint.
- *Existence.* Enforces that at a signal will eventually satisfy a certain property.

STL Net: Setup

- m signals over time $X = \{x^1, \dots, x^m\}$

- Subsequence of a signal $x_{[t,t']}^k$

- Model takes in “prefix” subsequences to determine the rest of the signal

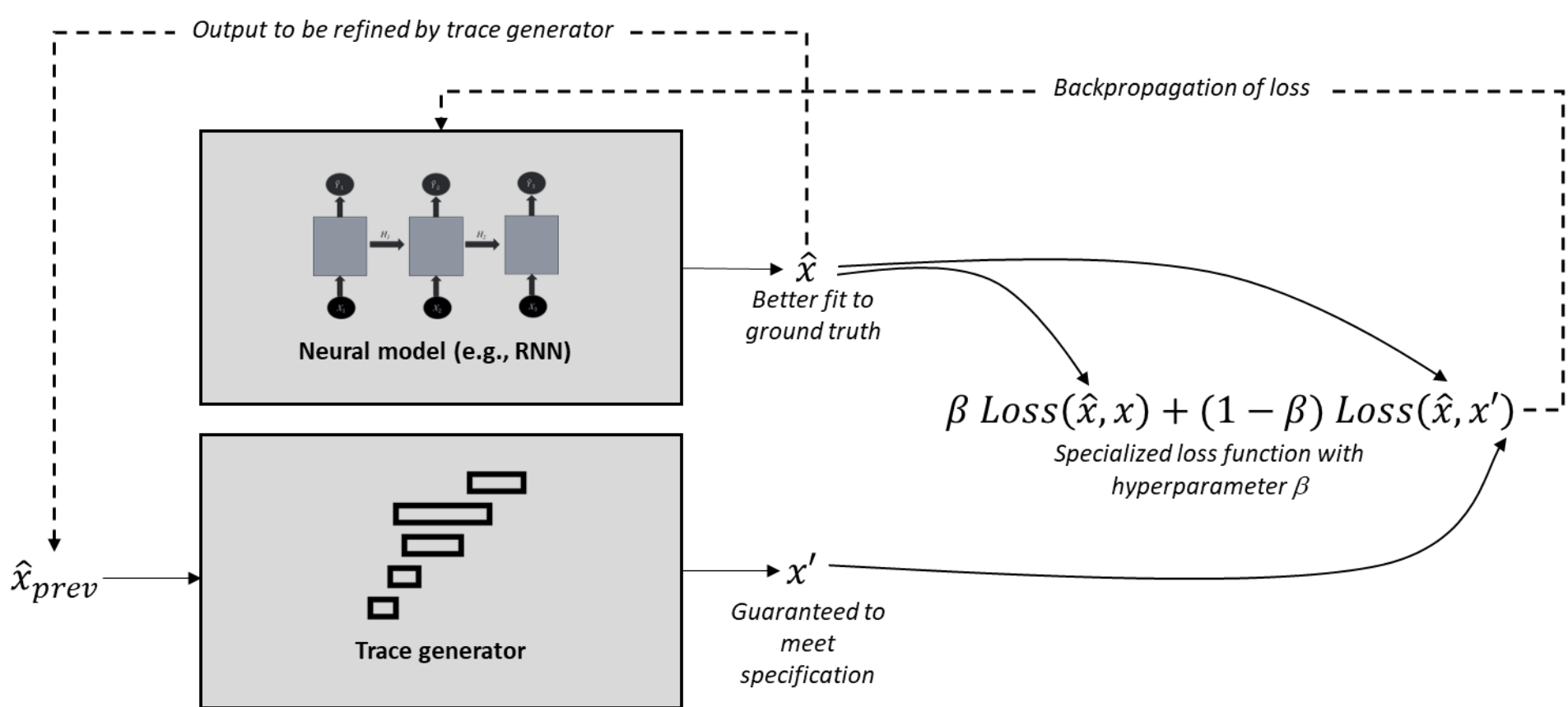
$$\Psi_{\theta}((x_{[1,i]}^1, \dots, x_{[1,i]}^m)) = (\hat{x}_{[i+1,n]}^1, \dots, \hat{x}_{[i+1,n]}^m)$$

- Goal: Find model parameters that both minimize loss and adhere to STL specifications

$$\theta = \mathbf{arg\ min}_{\theta} \mathbb{E}_{x \leftarrow X} [\mathcal{L}(\hat{x}, x)]$$

$$\text{such that } \hat{x} \models \varphi_1 \wedge \dots \wedge \varphi_v$$

Student-Teacher Framework



STL Loss Function

hyperparameter *ground truth* *teacher result*

$$\beta \mathcal{L}(\hat{x}, x) + (1 - \beta) \mathcal{L}(\hat{x}, x')$$

- The specialized loss function is used to compare the result from the neural network not only with the training data, but also with the result of the parent network
- Hyperparameter beta measures the trade-off between the two

Adjusting Neural Network Results to Meet a Specification

- We can think of the result of the neural model as a sequence of worlds over time (just note that the atoms in the worlds depend on analog values)
- If we can express a specification in a manner that allows us to simply compare worlds to the spec, we can update the trace in a straightforward manner

Key Idea: Converting Specification to DNF Form

- Turning the specification into DNF form (disjunction of conjunctions of literals) provides a few useful properties:
 1. Only one clause of the disjunction must be satisfied for the specification to be satisfied (so you can just iterate through clauses)
 2. The conjunctions of literals are very easy to compare with a world (as essentially you are just checking every atom and negation with each world in the sequence)
 3. Changing a sequence to meet a specification becomes trivial as you can simply modify the value associated with a specific predicate – and you can perform the modifications in such a way to be near the trace (via L1) as possible

Scalability

- STLNet relies the ability to convert STL formulas from CNF to DNF.
- However, doing so could result in an exponentially-large formula (Miltersen et al., 2005)
- Left alone, this could hinder the scalability of the approach, though unde

SATNet

Papers:

Wang, P., Donti, P.L., Wilder, B., Kolter, J.Z.: Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In: K. Chaudhuri, R. Salakhutdinov (eds.) Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, Proceedings of Machine Learning Research, vol. 97, pp. 6545–6554.

Chang, O., Flokas, L., Lipson, H., Spranger, M.: Assessing satnet's ability to solve the symbol grounding problem. In: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (eds.) Advances in Neural Information Processing Systems, vol. 33, pp. 1428–1439. Curran Associates, Inc. (2020).

Topan, S., Rolnick, D., Si, X.: Techniques for symbol grounding with satnet. Advances in Neural Information Processing Systems 34, 20733–20744 (2021)

The MAX SAT problem

INPUT:

Given Boolean variables x_1, \dots, x_m and clauses C_1, \dots, C_n where each clause is a disjunction of literals (atoms or negations). (Conjunctive normal form)

OUTPUT:

An assignment of Boolean variables such that the number of satisfied clauses is maximized.

Notes:

- Known to be NP-hard (even when each clause has just two literals)
- The clauses can be numerically represented by matrix (M) of dimensions $n \times m$ (in the original paper, S is used instead of M)
- Often framed as an optimization problem

Partial Knowledge MAX SAT variant

An extension to the problem is to partition the m Boolean variables into two groups: input $(a_{1,\dots,k}^{in})$ and output $(a_{k+1,\dots,m}^{out})$

Hence, we can think of MAX SAT as the following problem:





$$a_{k+1,\dots,m}^{out} = S(a_{1,\dots,k}^{in}, M)$$

Where S is an oracle and M is the matrix representing the clauses.

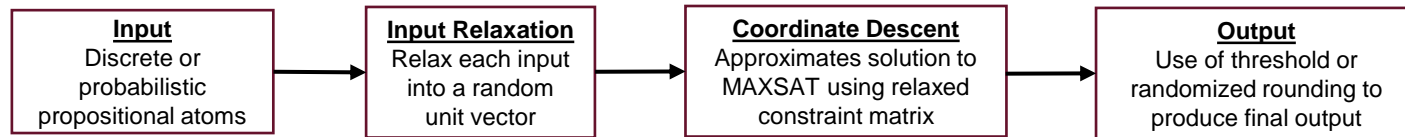
Sudoku can be framed as an instance of MAX SAT with partially known Boolean variables

<u>Input</u>			<u>Output</u>			
4			4	9	2	Sum to 15
3			3	5	7	Sum to 15
8		6	8	1	6	Sum to 15
			Sum to 15	Sum to 15	Sum to 15	

A visual variant of the problem is one in which the input is presented as images instead of text.

<u>Input</u>			<u>Output</u>			
			4	9	2	Sum to 15
			3	5	7	Sum to 15
			8	1	6	Sum to 15
			Sum to 15	Sum to 15	Sum to 15	

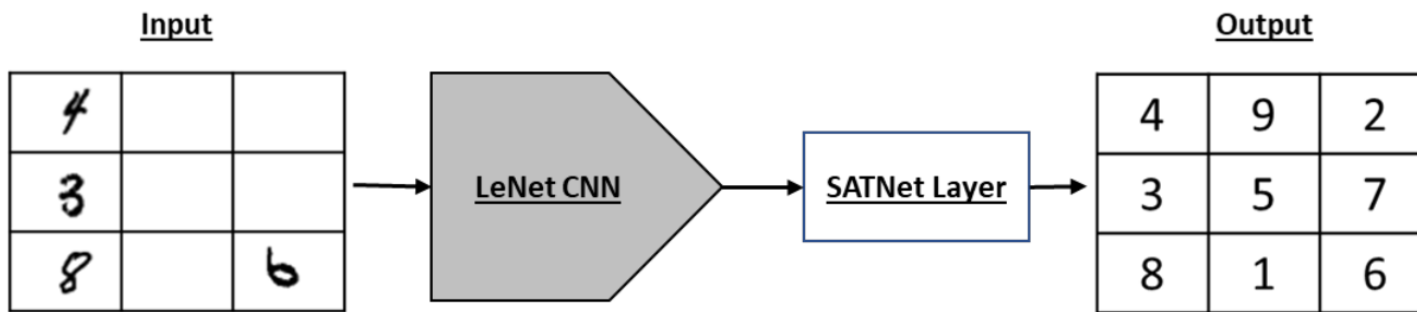
SAT Net Framework



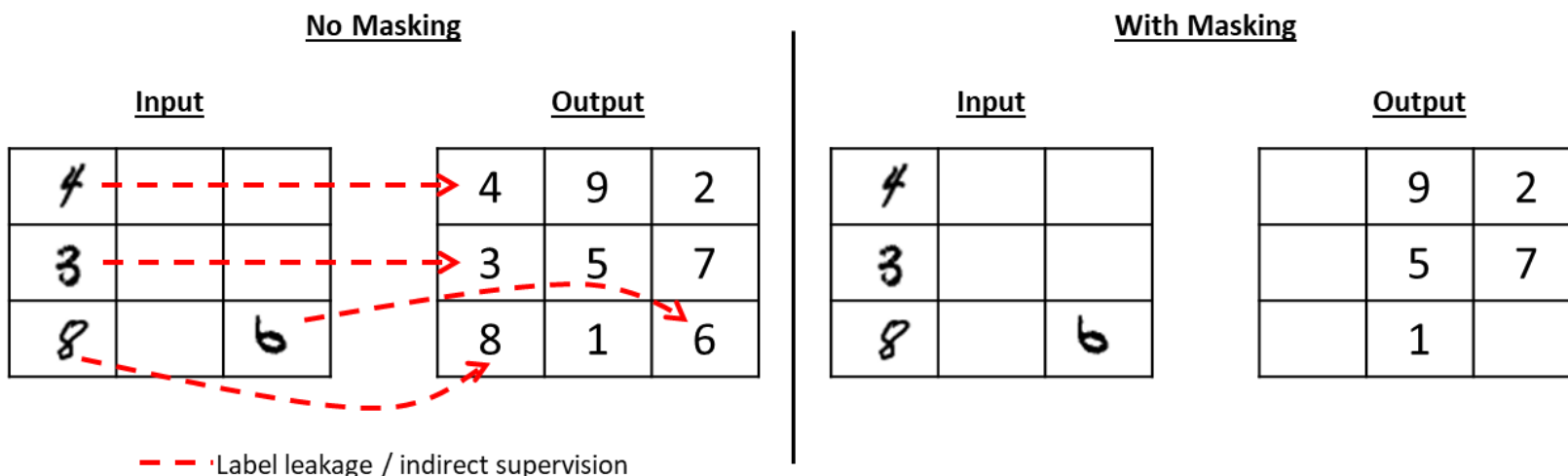
- A relaxation via Semi Definite programming is used to solve the MAX SAT instance
- The gradient is propagated through the SATNet layers using coordinate descent
- A relaxed constraint matrix is learned (i.e., for the approximate solution of the problem) as opposed to interpretable constraints
- 98.3% accuracy for Sudoku, 63.2% reported accuracy for visual case
- For the visual Sudoku problem, LeNet is used to classify the digits (pre-trained)

Visual Variant

- While standard CNN approach fails in visual case, SATNet approaches a theoretical limit (based on digit accuracy) for accuracy in visual Sudoku



- Later, Chang et al. (2020) showed that SATNet leveraged label leakage. SATNet fails catastrophically when labels are masked.



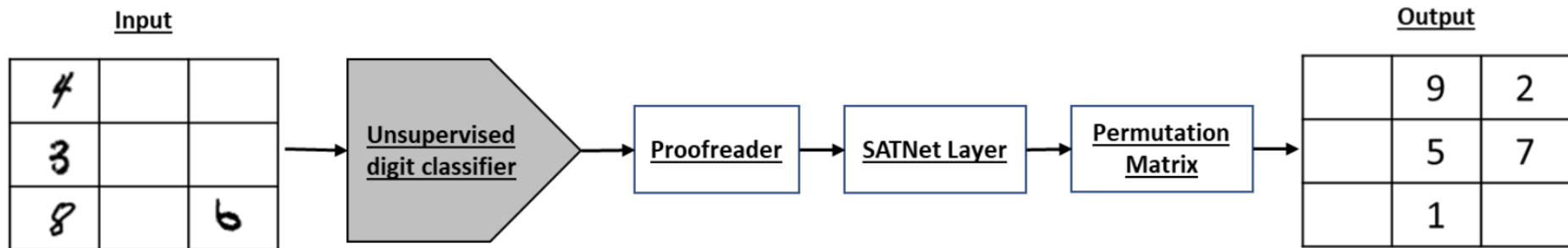
Despite its shortcomings, SAT Net has significant

- It successfully could learn constraints in a differentiable framework
- Combinatorial forward pass and ability to derive gradients for backpropagation
- Significantly outperformed standard DL architectures
 - (this was significant at the time of publication)

The relationship between symbols and perception

- Transduction problem
 - If they exist, how then, are the perceptual states mapped into amodal symbols? (Barsalou, 1999)
- Symbol grounding problem
 - The reverse of the transduction problem
 - How are amodal symbols grounded in perception? (Barsalou, 1999)
- Chang et al. argue that SAT Net did not adequately solve the symbol grounding problem.
 - They probably really mean the transduction problem – as the issue was the transduction of perception into symbols
 - Note: Symbol grounding does come up in ML verification – ensuring that a symbol maps back properly to perception

Unsupervised Learning to address transduction/symbol grounding



- Topan et al., seek to directly address the shortcomings of SAT Net:
 - Use of unsupervised learning for digit recognition
 - Additional loss term to account for inaccurate digits
 - Addition of proofreader layer improved performance (an extra boost, but not directly related to the problem of transduction)

More recent work

- Symbol grounding
 - Abduction (Dai & Muggleton, 2021), (Dai et al., 2019)
 - Appreciation / binarized neural networks (Evans et al., 2021)
 - DeepLogic (Duan, 2022)
- Learning constraints to combinatorial problems via deep learning
 - CombOptNet (Paulus et al., 2021)
 - Solver-Free (Nandawi et al., 2022)

Questions
