Safe Neurosymbolic Learning with Differentiable Symbolic Execution

Chenxi Yang, Swarat Chaudhuri
The University of Texas at Austin

ICLR

TEXAS
The University of Texas at Austin

Trishul
Trustworthy Intelligent Systems @ UT

Motivation

- Learning techniques are being used in safety-critical tasks.


Practical uses of Deep Neural Networks in Healthcare

deepakvraghavan May 2, 2018 · 9 min read


The Costly Pursuit of Self-Driving Cars Continues On. And On. And On.

Many in Silicon Valley promised that self-driving cars would be a common sight by 2021. Now the industry is resetting expectations and settling in for years of more work.

- In the real world, neural networks are often invoked by human-written code.

Provably Safe Neurosymbolic Learning

Safe Neurosymbolic Learning

- We focus on imitation learning.

Safe Neurosymbolic Learning

- We focus on imitation learning.
  - Human written code invoking neural networks

```
thermostat(x):
    N = 20
    isOn = 0.0
    i = 0
    while i < N:
        if isOn ≤ 0.5:
            isOn := π_θ^cool(x)
            x := COOLING(x)
        else:
            isOn, heat := π_θ^heat(x)
            x := WARMING(x, heat)
        i := i + 1
        assert(!EXTREME_TEMPERATURE(x))

    return
```

Safe Neurosymbolic Learning

- We focus on imitation learning.
    - Human written code invoking neural networks

```
thermostat(x):
    N = 20
    isOn = 0.0
    i = 0
    while i < N:
        if isOn ≤ 0.5:
            isOn := π_θ^cool(x)
            x := COOLING(x)
        else:
            isOn, heat := π_θ^heat(x)
            x := WARMING(x, heat)
        i := i + 1
        assert(!EXTREME_TEMPERATURE(x))

    return
```

2 different NNs

Safe Neurosymbolic Learning

- We focus on imitation learning.
  - Human written code invoking neural networks

Symbolic code

```
thermostat(x):
    N = 20
    isOn = 0.0
    i = 0
    while i < N:
        if isOn ≤ 0.5:
            isOn := π_θ^cool(x)
            x := COOLING(x)
        else:
            isOn, heat := π_θ^heat(x)
            x := WARMING(x, heat)
        i := i + 1
        assert(!EXTREME_TEMPERATURE(x))

    return
```

Safe Neurosymbolic Learning

- We focus on imitation learning
  - Human written code invoking neural networks
  - A trajectory dataset for imitation

```
thermostat(x):
    N = 20
    isOn = 0.0
    i = 0
    while i < N:
        if isOn ≤ 0.5:
            isOn := π_θ^cool(x)
            x := COOLING(x)
        else:
            isOn, heat := π_θ^heat(x)
            x := WARMING(x, heat)
        i := i + 1
        assert(!EXTREME_TEMPERATURE(x))

    return
```

20-length trajectories

Safe Neurosymbolic Learning

- We focus on imitation learning
  - Human written code invoking neural networks
  - A trajectory dataset for imitation
    - Trajectories are sequences of input-output pair of neural networks

```
thermostat(x):
    N = 20
    isOn = 0.0
    i = 0
    while i < N:
        if isOn ≤ 0.5:
            isOn := π_θ^cool(x)
            x := COOLING(x)
        else:
            isOn, heat := π_θ^heat(x)
            x := WARMING(x, heat)
        i := i + 1
        assert(!EXTREME_TEMPERATURE(x))

    return
```

Input-output pairs

Safe Neurosymbolic Learning

- We focus on imitation learning
    - Human written code invoking neural networks
    - A trajectory dataset for imitation
        - Trajectories are sequences of input-output pair of neural networks
        - In this example, trajectories are in the form of

```
<([x], [isOn], 'cool'), ([x], [isOn, heat], 'heat'), ([x], [isOn, heat], 'heat'), … >
```

Input-output pairs

```
thermostat(x):
    N = 20
    isOn = 0.0
    i = 0
    while i < N:
        if isOn ≤ 0.5:
            isOn := π_θ^cool(x)
            x := COOLING(x)
        else:
            isOn, heat := π_θ^heat(x)
            x := WARMING(x, heat)
        i := i + 1
        assert(!EXTREME_TEMPERATURE(x))

    return
```

Safe Neurosymbolic Learning

- We focus on imitation learning
  - Human written code invoking neural networks
  - A trajectory dataset for imitation
    - Trajectories are sequences of input-output pair of neural networks
    - In this example, trajectories are in the form of

Input of NN

<([x], [isOn], 'cool'), ([x], [isOn, heat], 'heat'), ([x], [isOn, heat], 'heat'), … >

```
thermostat(x):
    N = 20
    isOn = 0.0
    i = 0
    while i < N:
        if isOn ≤ 0.5:
            isOn := π_θ^cool(x)
            x := COOLING(x)
        else:
            isOn, heat := π_θ^heat(x)
            x := WARMING(x, heat)
        i := i + 1
        assert(!EXTREME_TEMPERATURE(x))

    return
```

Input-output pairs

Safe Neurosymbolic Learning

- We focus on imitation learning
  - Human written code invoking neural networks
  - A trajectory dataset for imitation
    - Trajectories are sequences of input-output pair of neural networks
    - In this example, trajectories are in the form of

Input of NN

Output of NN

```
<([x], [isOn], 'cool'), ([x], [isOn,
heat], 'heat'), ([x], [isOn, heat],
'heat'), … >
```

```
thermostat(x):
    N = 20
    isOn = 0.0
    i = 0
    while i < N:
        if isOn ≤ 0.5:
            isOn := π_θ^cool(x)
            x := COOLING(x)
        else:
            isOn, heat := π_θ^heat(x)
            x := WARMING(x, heat)
        i := i + 1
        assert(!EXTREME_TEMPERATURE(x))

    return
```

Input-output pairs

Safe Neurosymbolic Learning

- We focus on imitation learning
  - Human written code invoking neural networks
  - A trajectory dataset for imitation
    - Trajectories are sequences of input-output pair of neural networks
    - In this example, trajectories are in the form of

Input of NN

Output of NN

```
<([x], [isOn], 'cool'), ([x], [isOn,
heat], 'heat'), ([x], [isOn, heat],
'heat'), … >
```

Label of NN

```
thermostat(x):
    N = 20
    isOn = 0.0
    i = 0
    while i < N:
        if isOn ≤ 0.5:
            isOn := π_θ^cool(x)
            x := COOLING(x)
        else:
            isOn, heat := π_θ^heat(x)
            x := WARMING(x, heat)
        i := i + 1
        assert(!EXTREME_TEMPERATURE(x))

    return
```

Input-output pairs

Safe Neurosymbolic Learning

- We focus on imitation learning.
  - Human written code invoking neural networks
  - A trajectory dataset for imitation
  - Constraints on the input

```
thermostat(x):
    N = 20
    isOn = 0.0
    i = 0
    while i < N:
        if isOn ≤ 0.5:
            isOn := π_θ^cool(x)
            x := COOLING(x)
        else:
            isOn, heat := π_θ^heat(x)
            x := WARMING(x, heat)
        i := i + 1
        assert(!EXTREME_TEMPERATURE(x))

    return
```

$x \in [55.0,\ 70.0]$

Safe Neurosymbolic Learning

- We focus on imitation learning
  - Human written code invoking neural networks
  - A trajectory dataset for imitation
  - Constraints on the input
  - Constraints on the trajectory of the system

```
thermostat(x):
    N = 20
    isOn = 0.0
    i = 0
    while i < N:
        if isOn ≤ 0.5:
            isOn := π_θ^cool(x)
            x := COOLING(x)
        else:
            isOn, heat := π_θ^heat(x)
            x := WARMING(x, heat)
        i := i + 1
        assert(!EXTREME_TEMPERATURE(x))

    return
```

Safety constraints

Formulation

Fitting the trajectory dataset
Data Signal: $Q(\theta)$

Satisfying the safety constraint
Safety Signal: $C(\theta)$

$$\boxed{\min_{\theta} Q(\theta) \qquad \text{s.t.} \;\; C(\theta) \leq 0.}$$

Safe Learning Framework

Fitting the trajectory dataset
Data Signal: $Q(\theta)$

Satisfying the safety constraint
Safety Signal: $C(\theta)$

$$\boxed{\min_{\theta} Q(\theta) \qquad \text{s.t.} \;\; C(\theta) \leq 0.}$$

Using Lagrange multipliers

$$L(\theta, \lambda) = Q(\theta) + \lambda C(\theta)$$

High-level: the framework repeatedly solves this optimization problem

$$\min_{\theta} Q(\theta) + \lambda \boxed{C(\theta)}$$

Challenge: Estimate the gradient of safety signal in a differentiable way.

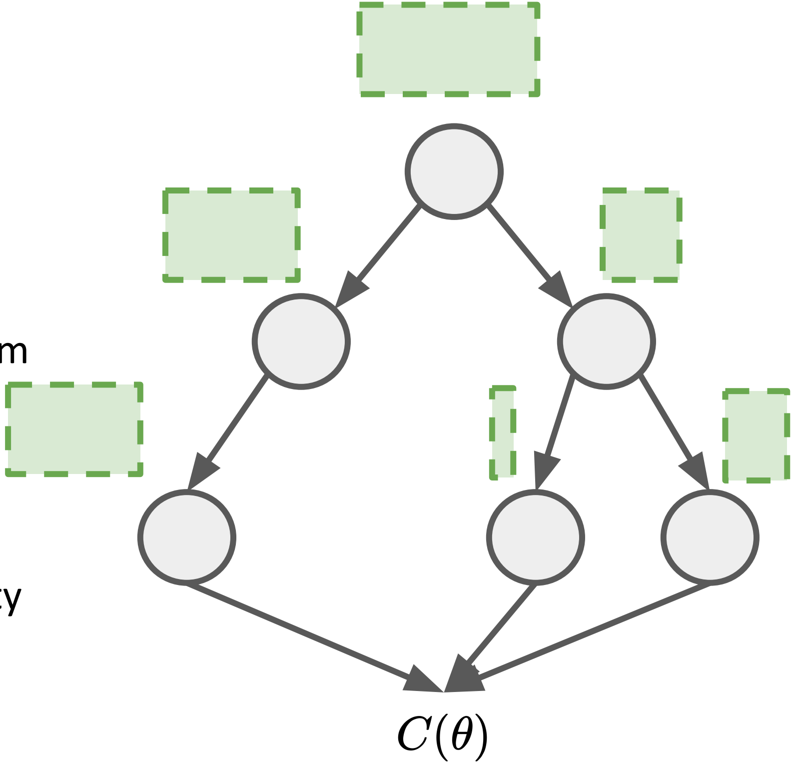
Verification Technique: Symbolic Execution

! Two Issues

Symbolic input

Propagate the symbolic input through the program

Aggregate the safety measurement from all paths



Path explosion: Branch X Loops

Non-differentiability: Discrete branches

DSE: Differentiable Symbolic Execution

Volume weighted probability

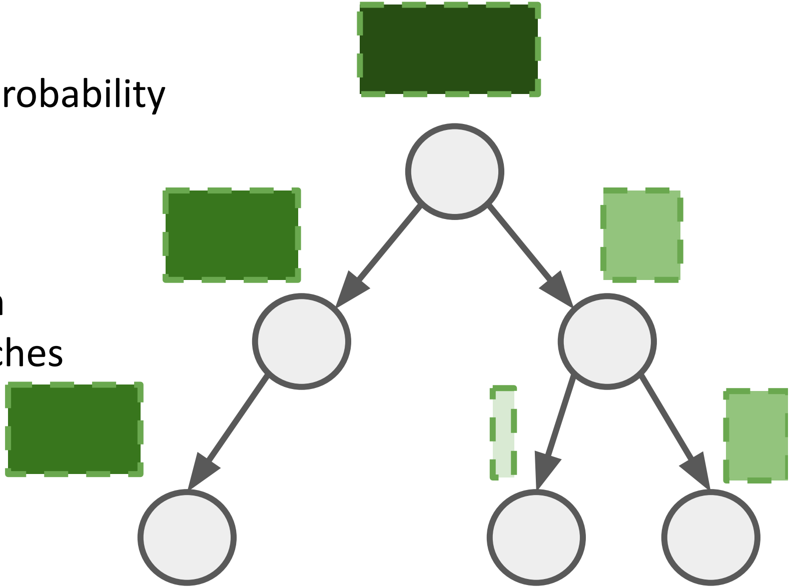
$$p_\theta(t_j|\sigma_i) = \frac{\boxed{\mathbf{Vol}(G_j \wedge V_i)}}{\boxed{\mathbf{Vol}(V_i)}}$$

The volume of the symbolic state that satisfies G.

The entire volume of the symbolic state before the conditional.

Sample paths when encountering branches

$$p_\theta(\tau_\theta^{\#}) = p(\sigma_0) \prod_i p_\theta(t_i|\sigma_{i-1}).$$



Use symbolic REINFORCE to estimate the gradient of safety loss.

$$C^{\#}(\theta) = \mathbf{E}_{\tau^{\#} \sim p_\theta(\tau^{\#})}\, Unsafe_\theta(\tau^{\#}).$$

DSE: Differentiable Symbolic Execution

Estimate the gradient

$$
\begin{aligned}
& \nabla_\theta(C^\#(\theta)) \\
= \quad & \nabla_\theta \mathbf{E}_{\tau^\# \sim p_\theta(\tau^\#)} Unsafe_\theta(\tau^\#) \\
= \quad & \boxed{\mathbf{E}_{\tau^\# \sim p_\theta(\tau^\#)}[\nabla_\theta Unsafe_\theta(\tau^\#)]} + \boxed{\mathbf{E}_{\tau^\# \sim p_\theta(\tau^\#)}[Unsafe_\theta(\tau^\#)\nabla_\theta(\log p_\theta(\tau^\#))]}
\end{aligned}
$$

Penalty is a function over NN's parameter
Exhibit the symbolic unsafety change of a path

Follow the classic REINFORCE estimator
Exhibit the probability change of a symbolic path

Safe Learning Framework

Data Signal: $Q(\theta)$
Fitting the dataset

Safety Signal: $C(\theta)$
Satisfying the safety constraint

$$\boxed{\min_{\theta} Q(\theta) \qquad \text{s.t. } C(\theta) \leq 0.}$$

$$\min_{\theta} Q(\theta) + \lambda C(\theta)$$

Challenge: Estimate the gradient of safety signal in a differentiable way.

Safe Learning Framework

Data Signal: $Q(\theta)$
Fitting the dataset

Safety Signal: $C(\theta)$
Satisfying the safety constraint

$$\boxed{\min_{\theta} Q(\theta) \qquad \text{s.t. } C(\theta) \leq 0.}$$

$$\min_{\theta} Q(\theta) + \lambda \boxed{C^{\#}(\theta)}$$

Approximate Safety Loss

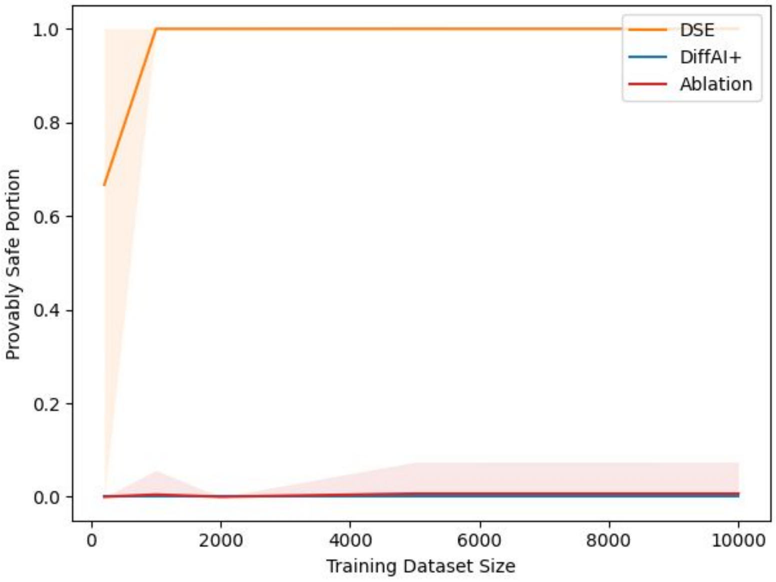
Differentiable **S**ymbolic **E**xecution
(**DSE**)

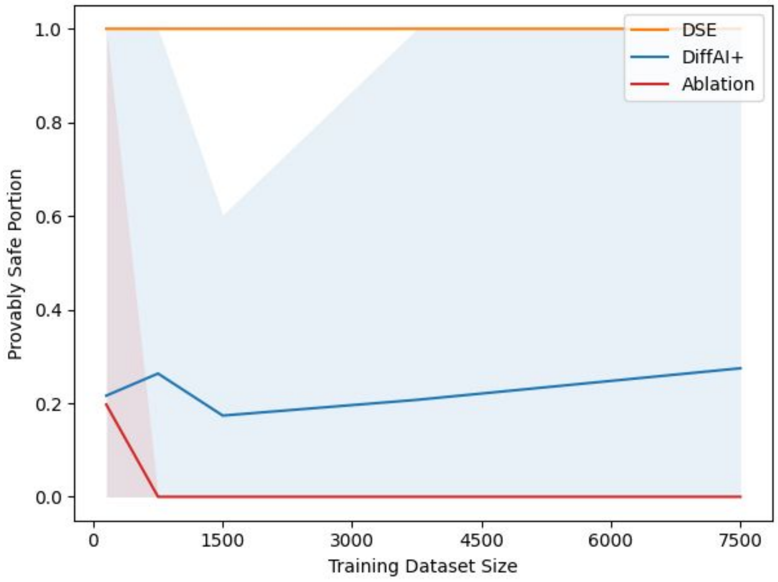
Experiments

Four benchmarks

- Thermostat: 2 NN controllers and ~200 dynamic lines of code
- Racetrack: 2 competing vehicles controlled by NN and interactions with the map
- Aircraft collision: 1 aircraft NN controller and 4^15 paths in maximum
- Cartpole: 1 cart pole controlled by NN

Evalution





The larger y, the safer learnt progra. DSE performs much better than two baselines: Ablation and DiffAI+.

Summary

- We provide DSE, a differentiable way to combine verification techniques over symbolic code with safe learning.
- In practice, DSE can learn provably safer programs than SOTA.

Thank you!

Paper: http://arxiv.org/abs/2203.07671
Code: https://github.com/cxyang1997/DSE