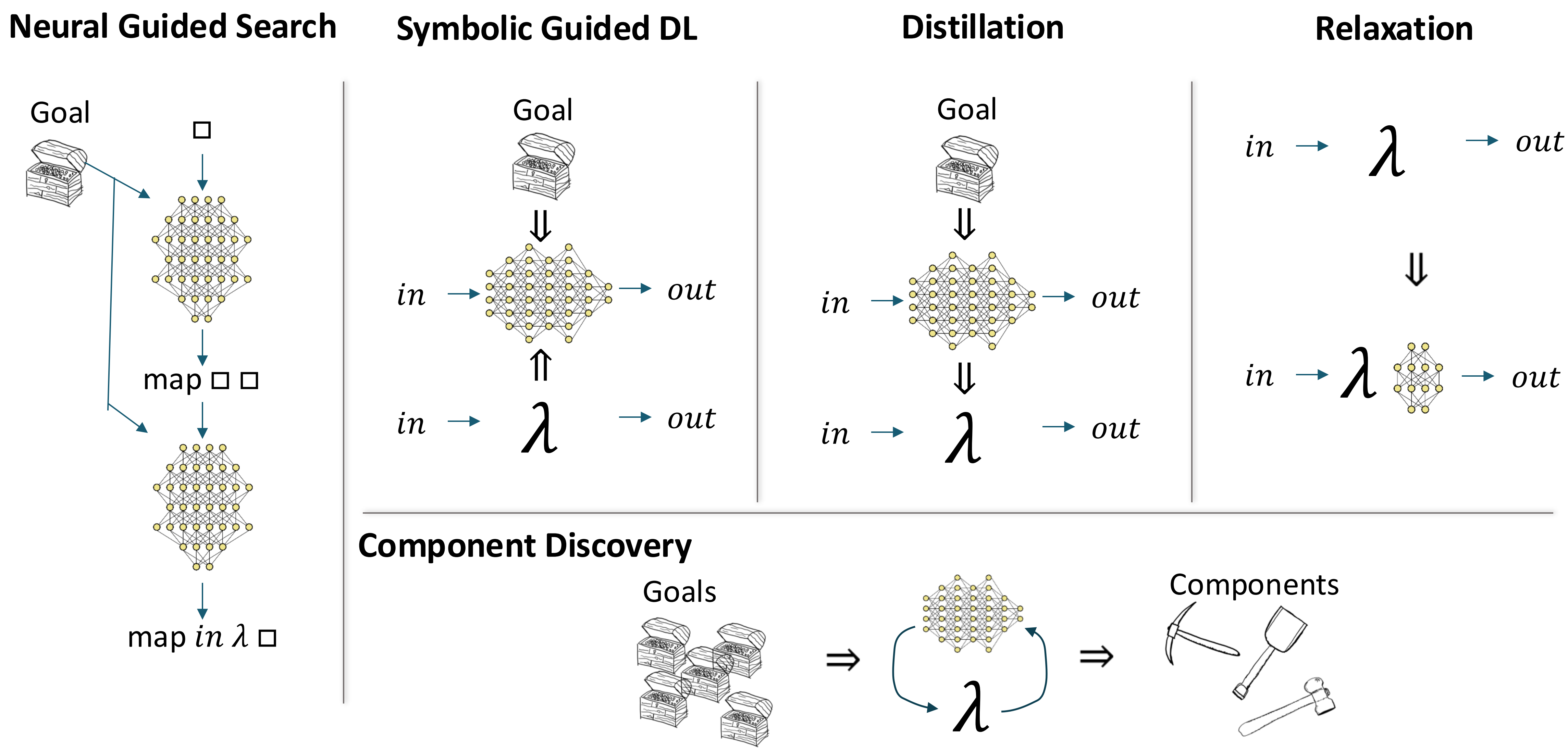
Lecture 36
Neurosymbolic Synthesis - I

Sankha Narayan Guria

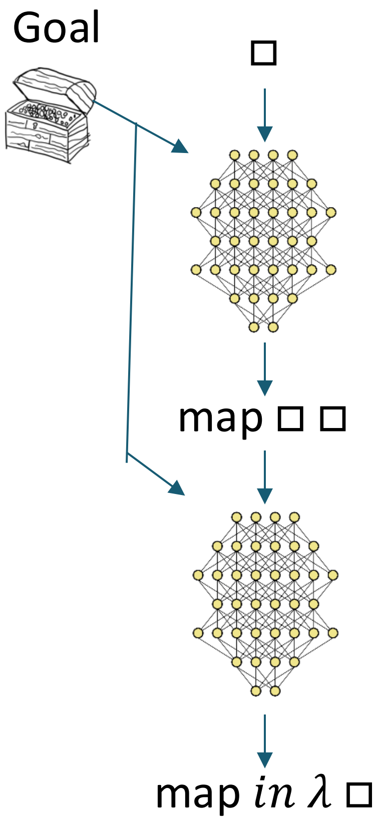
with slides from

Armando Solar-Lezama in-turn based on slides from Swarat Chaudhuri

Neurosymbolic program synthesis



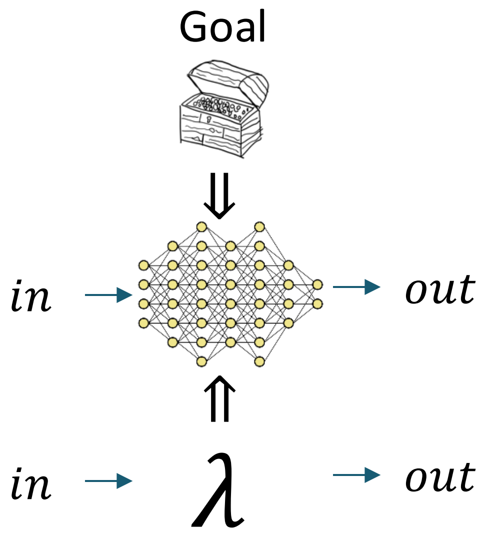
Neural Guided Search



Leverage the ability of NN to learn complex conditional distributions

Network guides the search for programs that satisfy the goals

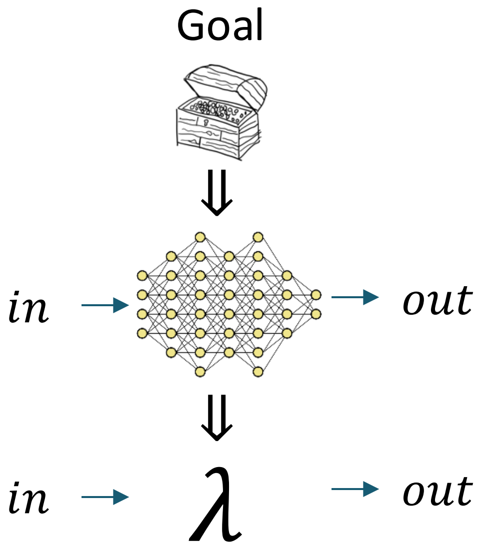
Symbolic Guided Deep Learning



Symbolic knowledge can be used to guide the training of neural networks

- When you want a network that is consistent with prior knowledge
- When you want to improve data efficiency and better generalization

Distillation

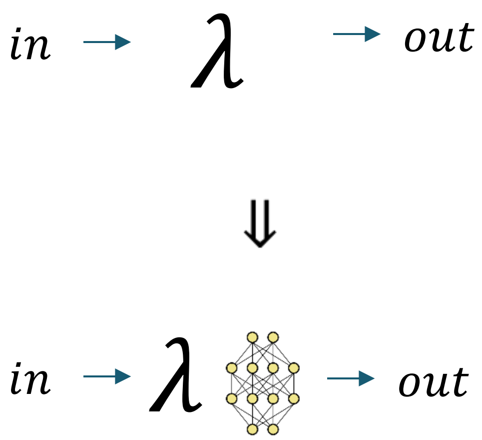


Use the neural network as a starting point for program synthesis

Replace neural components with symbolic ones

- To improve interpretability and analyzability
- To better generalize out of distribution
- To ensure more predictable behavior

Relaxation



Replace symbolic components with neural proxies

- Can help leverage the information in the symbolic component into a larger DL pipeline
- Can help guide the search for symbolic components

Relaxation:
Supporting numerical optimization over discrete programs

The Parameter Synthesis Problem

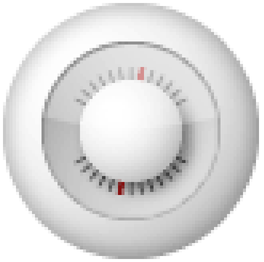
```
tOff := ??;    tOn := ??;  h := ??
forever{
   temp := readTemp();
   if (isOn() && temp > tOff)
        switchHeaterOff();
  elseif ( !isOn() && temp < tOn)
        switchHeaterOn(h);
}
```

Warming:

$$\frac{d}{dt} temp = -k \cdot temp + h$$

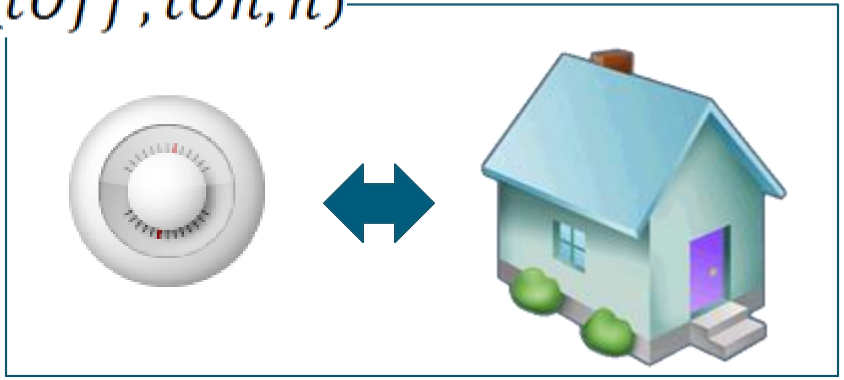
Cooling:

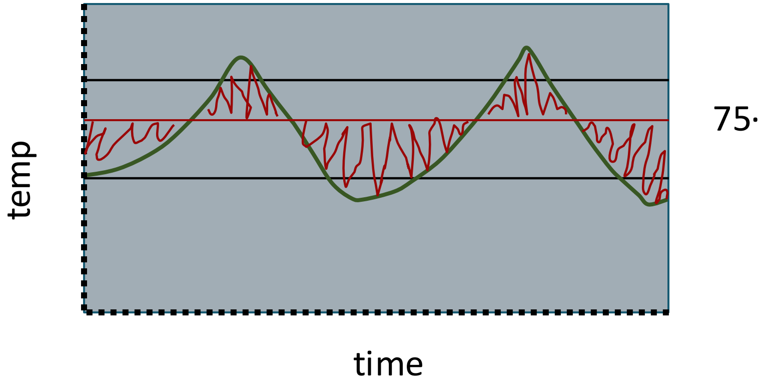
$$\frac{d}{dt} temp = -k \cdot temp$$



The Parameter Synthesis Problem

$F(tOff, tOn, h)$





$$F(tOff, tOn, h) = [\, t_0, t_1, \ldots, t_k]$$

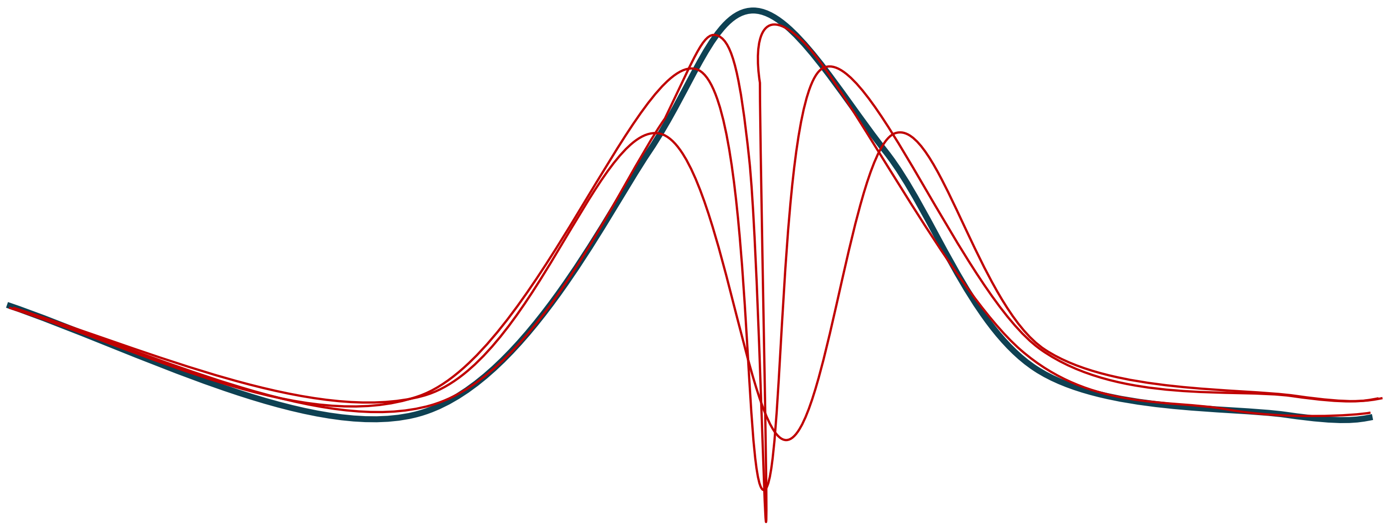
$$Err(tOff, tOn, h) := \sum (t_i - 75)^2$$

Can we find values of (tOff, tOn, h) to minimize Err?

Limitations

Smooth function converges to original function according to L2 norm.

- That's not enough for optimization.
- The function below converges to the original function, but the local minima is in the worst possible point



Limitations

Programs tend to be difficult to optimize

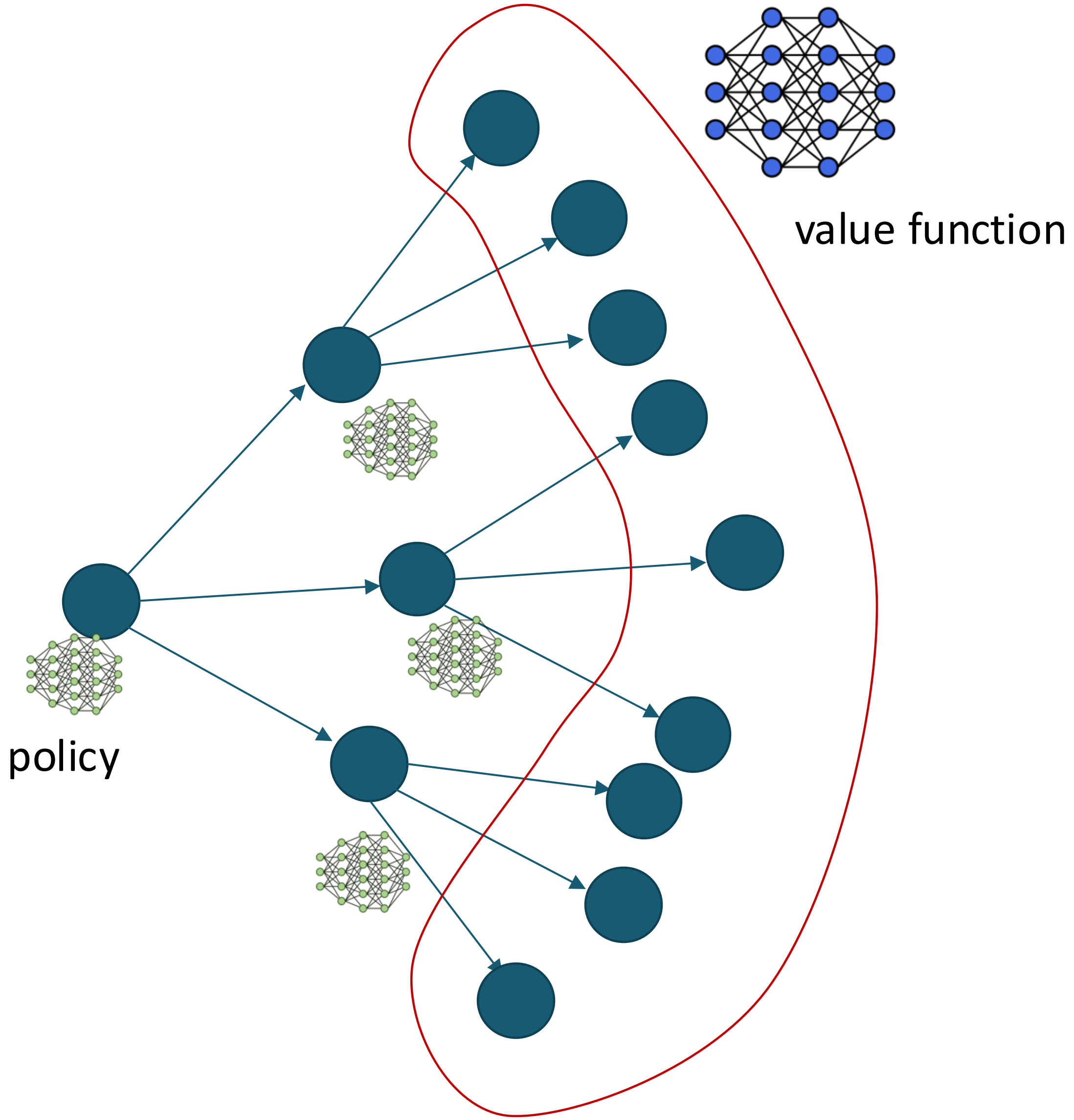
- Even with smoothing, there are too many local minima

Can be useful in combination with other search techniques

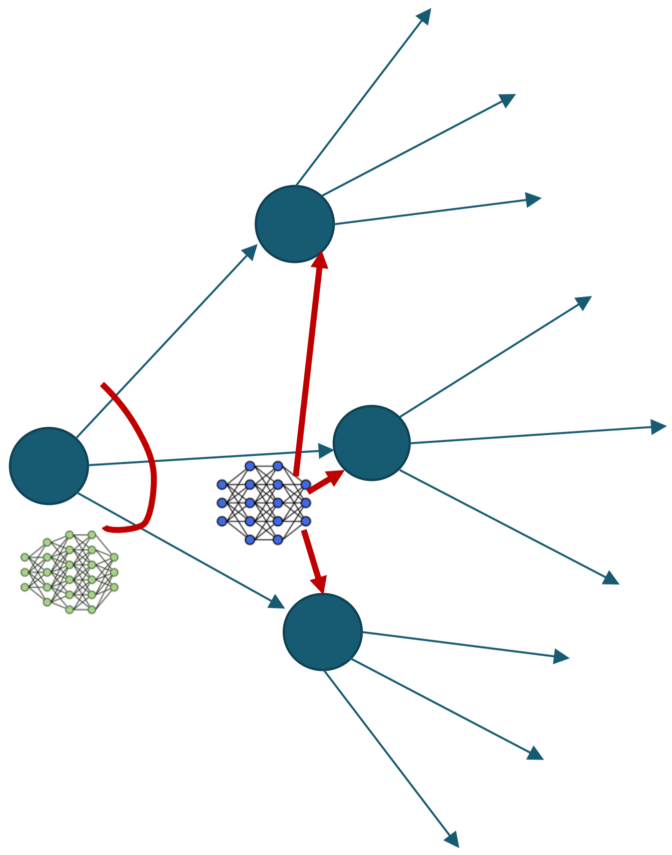
- e.g. Hamiltonian Monte Carlo

Relaxation + Top-Down = Near

Learning to synthesize incrementally 2



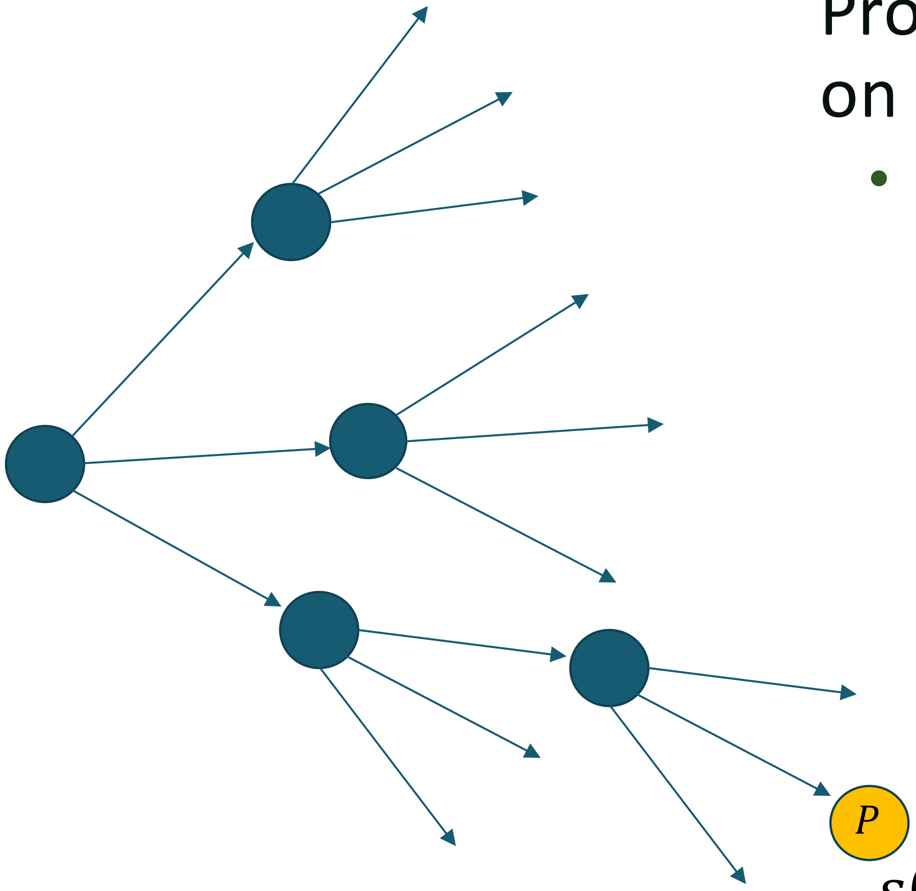
Guiding program search



So far

- Neural network policy to guide search
- Value function scores individual states

Guiding program search

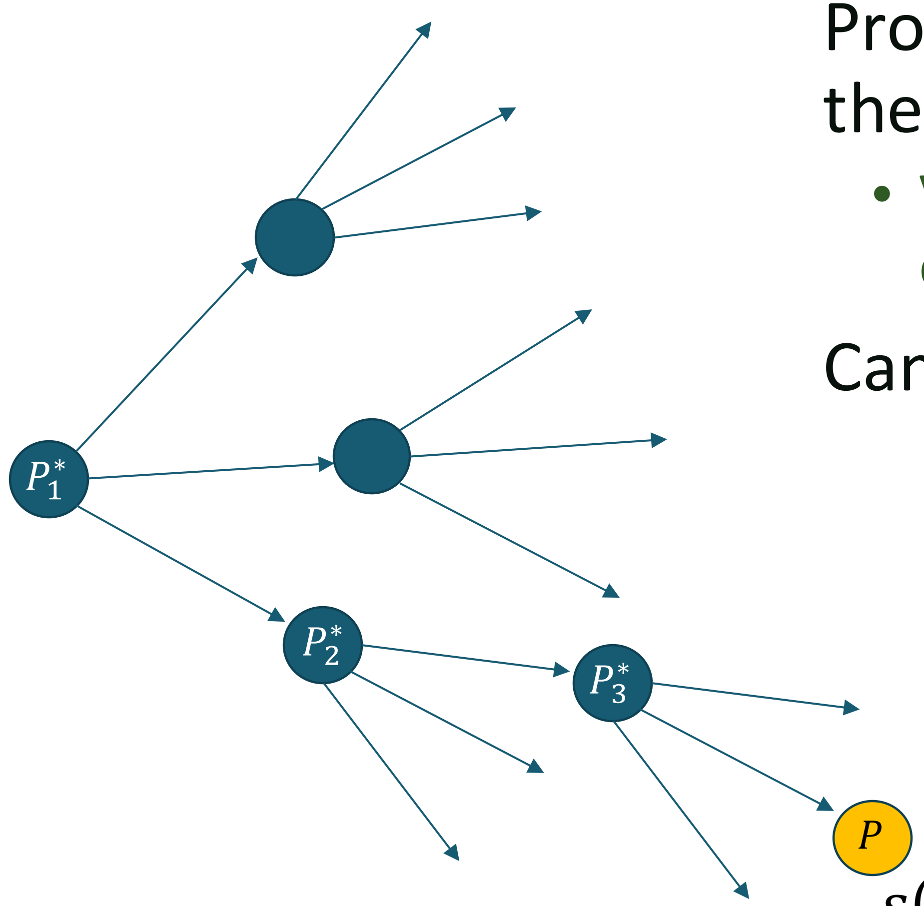


$$s(P) + \min_{\theta} \mathrm{Loss}(\alpha_P, \theta_P)$$

Problem: You only get ground truth on the leaves of the search tree

- Value for an intermediate node is only an estimate

Guiding program search



Problem: You only get ground truth on the leaves of the search tree

- Value for an intermediate node is only an estimate

Can we get a better estimate with DL?

$$s(P) + \min_{\theta} \text{Loss}(\alpha_P, \theta_P)$$

Estimating the "Cost to Go"

P^* = partial program (non-terminal nodes)

$P^*[? \rightarrow e]$ = completions of P^* (reachable terminal nodes)

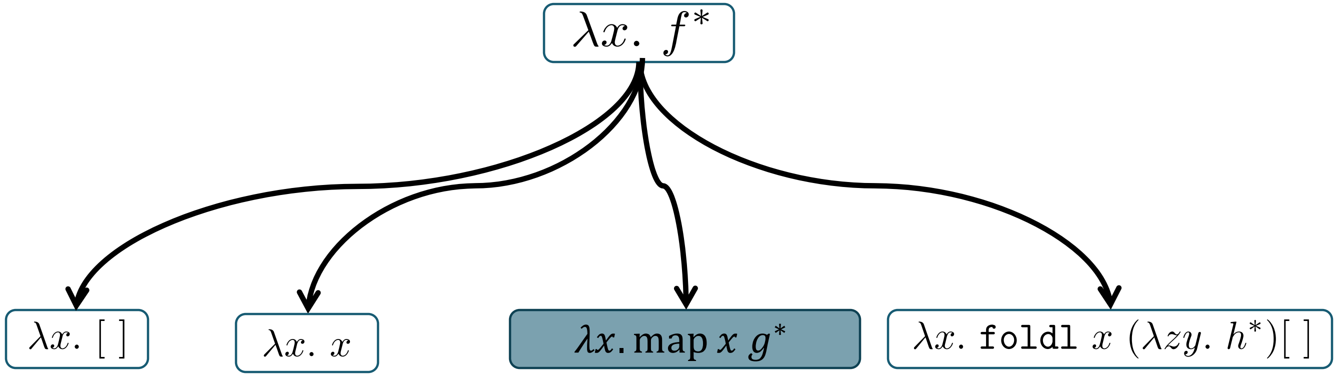
$$\text{Heuristic Estimate: } d(P^*) \approx \overbrace{\min_{P \in \mathbb{C}(P^*)} \left[\underbrace{\Delta s(P, P^*)}_{\text{Additional Structure Cost}} + \underbrace{\min_{\theta} \mathrm{Loss}(\alpha_P, \theta_P)}_{\text{Training Loss}} \right]}^{\text{"Cost to Go"}}$$

If $d(P^*)$ is a lower bound it becomes an "admissible heuristic"

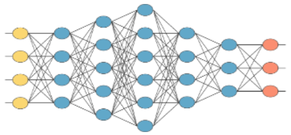
Learning with Neural Heuristics

Shah et al. ***Learning Differentiable Programs with Neural Admissible Heuristics.*** NeurIPS 2020.

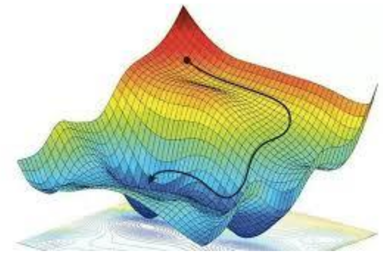
Guiding Search with Neural Relaxations



g^*



Fill hole with NN



Train parameters

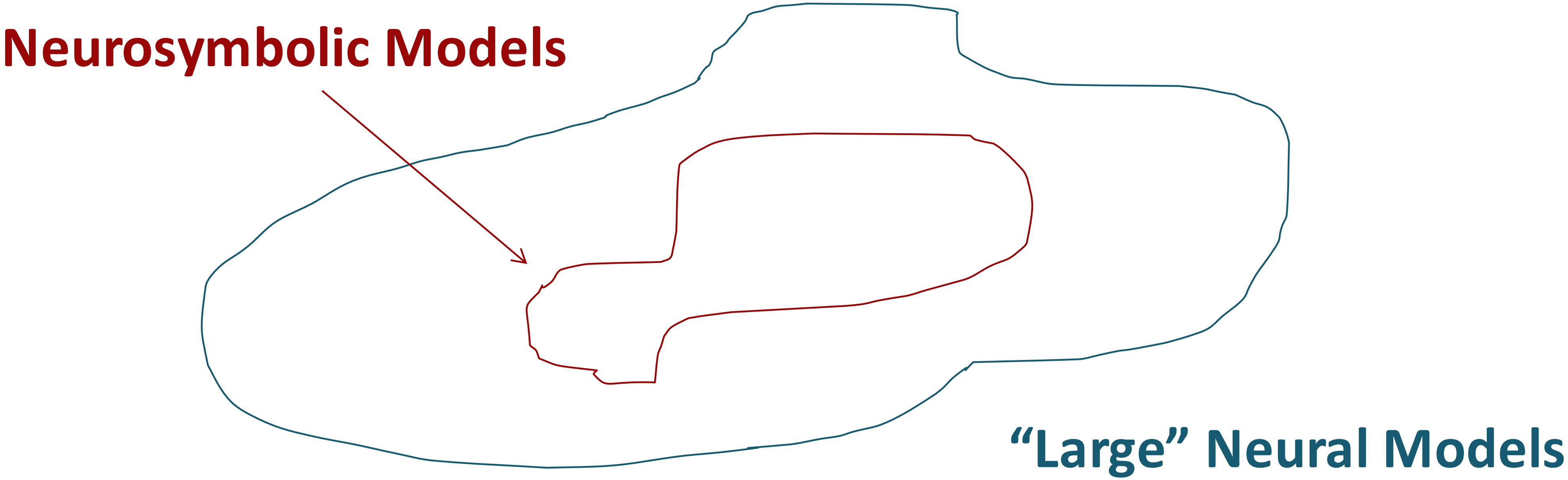
Use training loss to guide search

If a large neural network cannot fit this hole, then a neurosymbolic completion also cannot

Learning Differentiable Programs with Admissible Neural Heuristics, Ameesh Shah*, Eric Zhan*, et al., NeurIPS 2020

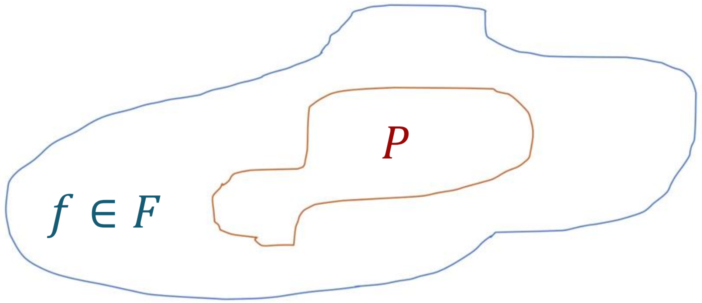
Motivating Observation/Assumption:

~~Functional Representational~~ Power



“Neural Relaxation” Every neurosymbolic model can be (approximately) represented by some “large” neural model.

Implication
(abstract form)



Large Neural

Slack due to approximation error
or training ability

$$\forall P, \exists f \in F \text{ s.t. } d(f) \leq d(P) + \epsilon$$

Neurosymbolic

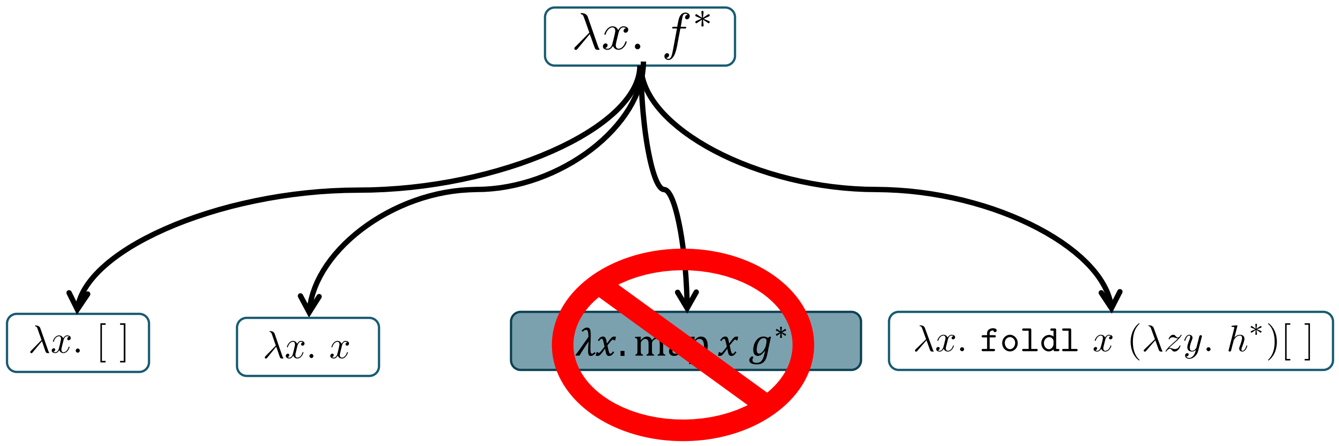
Any Cost Function

We can train an admissible heuristic!

“Neural Relaxation” Every neurosymbolic model can be (approximately) represented by some “large” neural model.

Informed Search (e.g., A*)

Use $d(P^*)$ to prune the search



Can Prune This Branch!

Suppose:

Structural Cost

Training Loss

$$s(\boxed{\lambda x. map\ x\ g^*}) + d(\boxed{\lambda x. map\ x\ g^*}) > s(\boxed{\lambda x.\ x}) + \text{Loss}(\boxed{\lambda x.\ x})$$

"Cost to Go" Heuristic

A* Search

Priority queue of current leaf nodes:

- Sorted by $s(P^*) + d(P^*)$

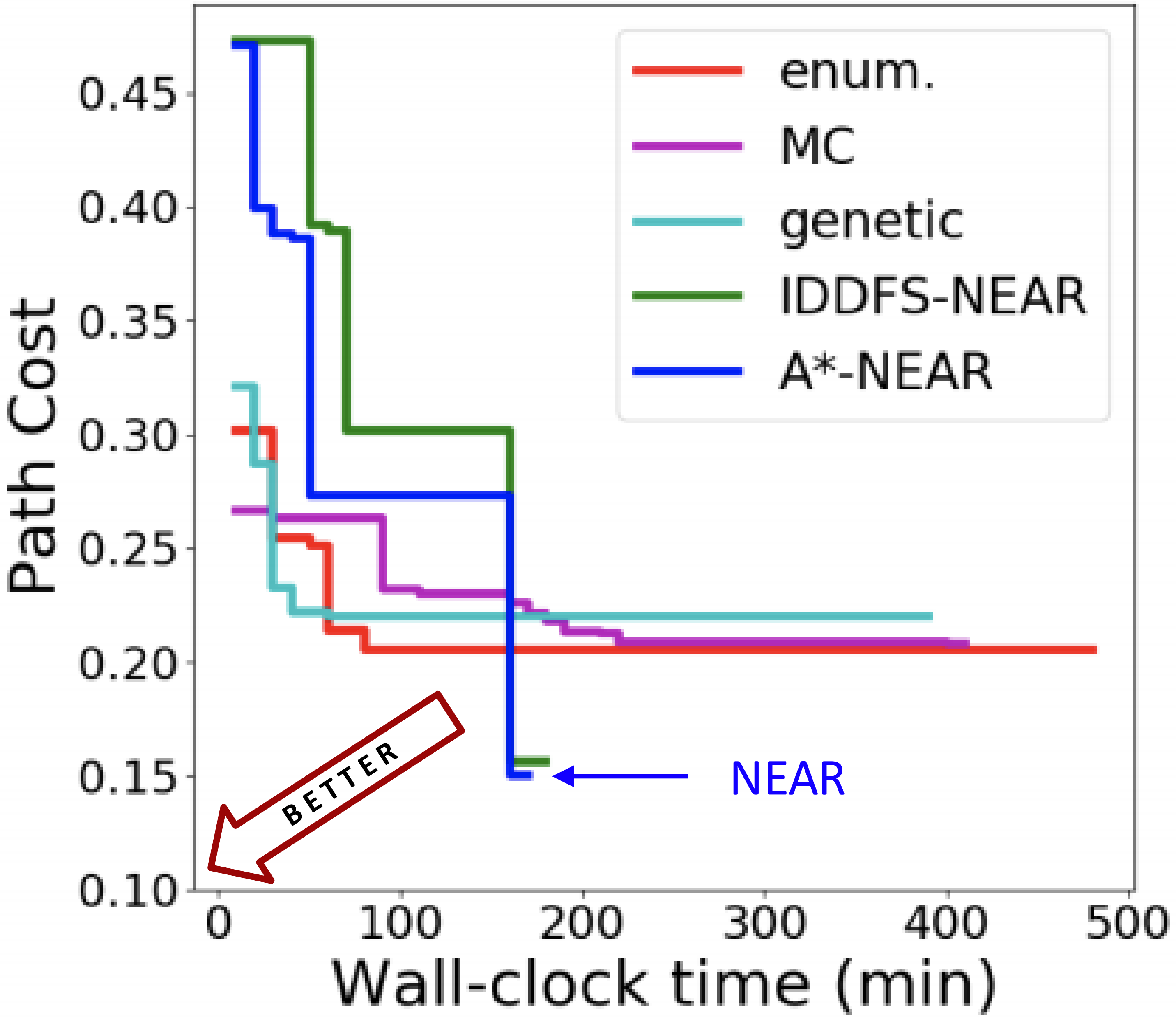
Pop off top program P^*

- If P^* is complete, terminate
- Else, expand P^*, add child nodes to priority queue

Guarantee: if $d(P^*)$ is $\underbrace{\text{admissible}}_{\text{Lower bounds “Cost to Go”}}$, A* will return optimal P

- Tighter $d(P^*)$ prunes more aggressively
- Uninformed $d(P^*)$ (e.g., always 0) => uninformed search

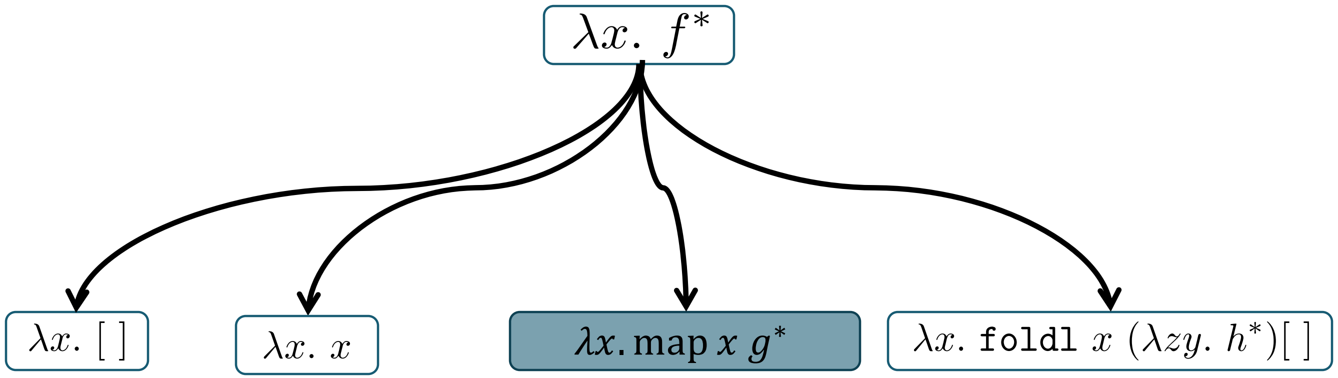
NEAR: Results



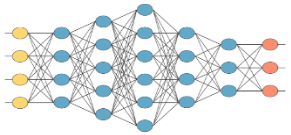
Order of magnitude speedup!

Learning Differentiable Programs with Admissible Neural Heuristics, Ameesh Shah*, Eric Zhan*, et al., NeurIPS 2020

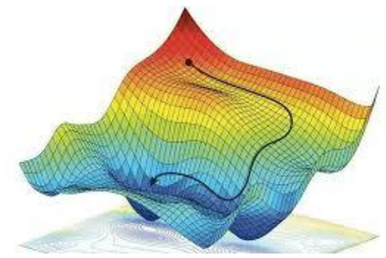
NEAR: Neural Admissible Relaxations



g^*



Fill hole with NN



Train parameters

Use training loss as admissible heuristic

If a large neural network cannot fit this hole, then a neurosymbolic completion also cannot

Learning Differentiable Programs with Admissible Neural Heuristics, Ameesh Shah*, Eric Zhan*, et al., NeurIPS 2020