Learning Navigation Costs from Demonstration via Differentiable Planning

Tianyu Wang Vikas Dhiman Nikolay Atanasov

UC San Diego

Presenter: Yuanhang Zhang

Motivation

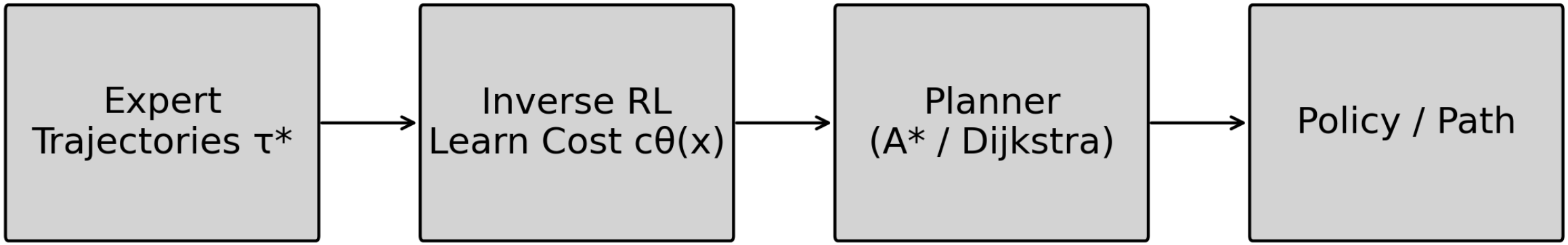
- Autonomous robots must operate in unstructured, and dynamically changing environments.
- The true cost function encoding safe, dynamically feasible, and efficient behavior is generally not available.
- Humans and animals navigate successfully with partial knowledge, adapting to new obstacles from prior experience.
- Expert demonstrations (sequences of observations, states, and controls) are available — but direct queries of the cost function are not.

> **Key Insight:**
>
> We need to infer the cost function from expert behavior and partial observations, not engineer it by hand.

Preliminary: Classic Inverse Reinforcement Learning (IRL)

- **Method**: Given expert demonstrations, recover the underlying cost. Assumes **fully observable** and **fixed maps**.

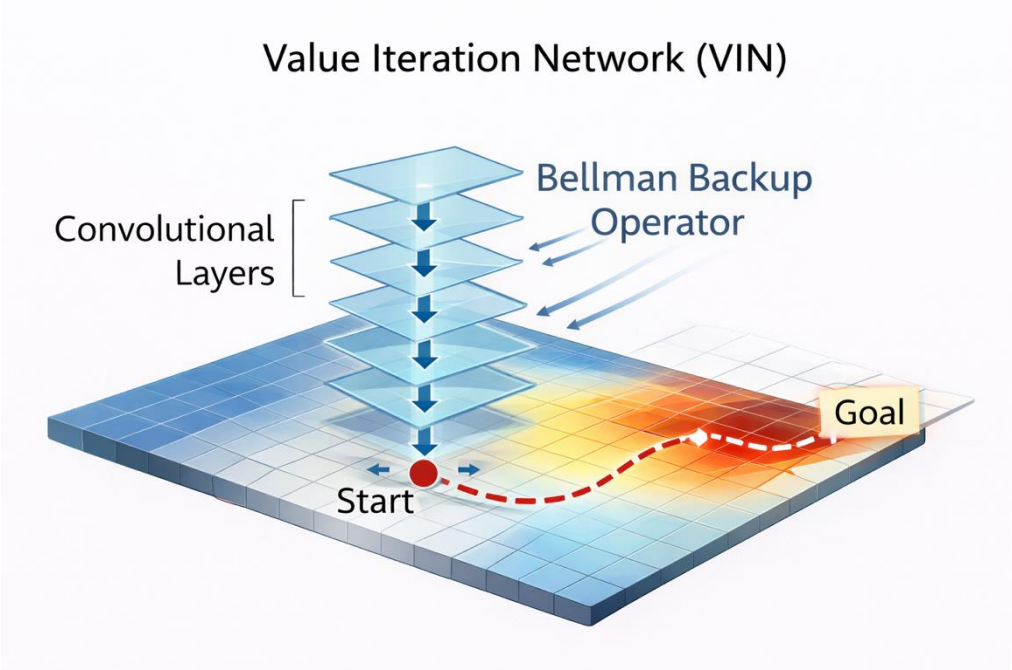
- **Key Challenge**: The planner must be **differentiable** so policy loss can backpropagate into cost that we parameterize.

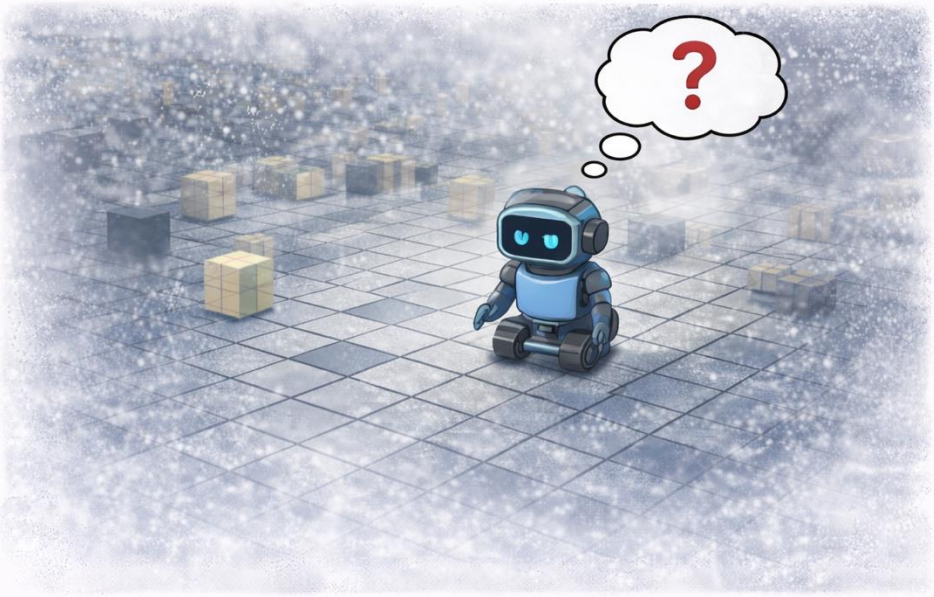


Prior Work: Value Iteration Networks (VIN)

- **Method**: Performs value iteration **over the entire state space** using convolutional layers as Bellman backup operators.

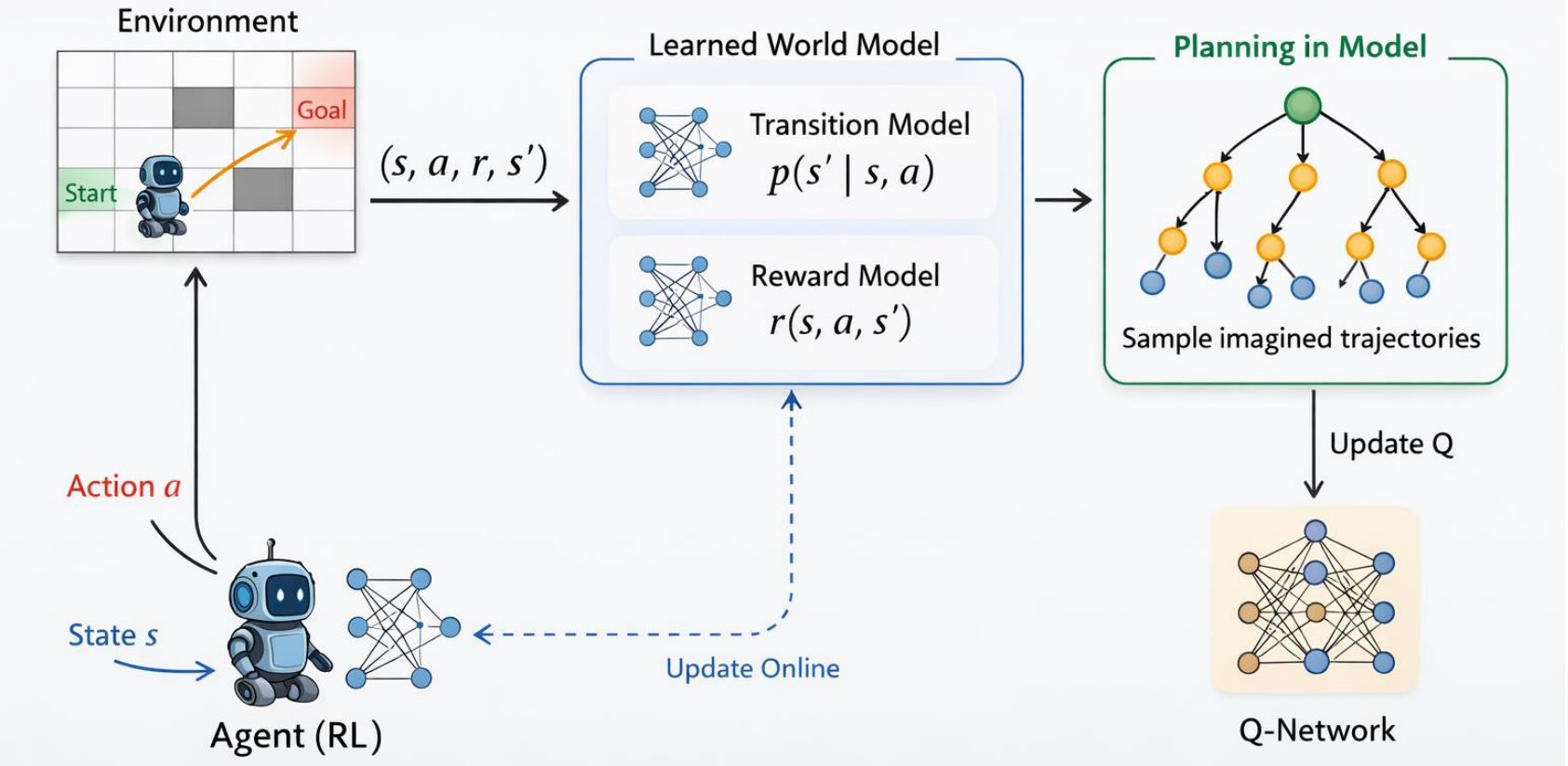
- **Weakness**: Computing values for **all states is inefficient** for large grids or complex environments with high-dimensional states. It does **not naturally handle partial or noisy observations**, which is often infeasible online.





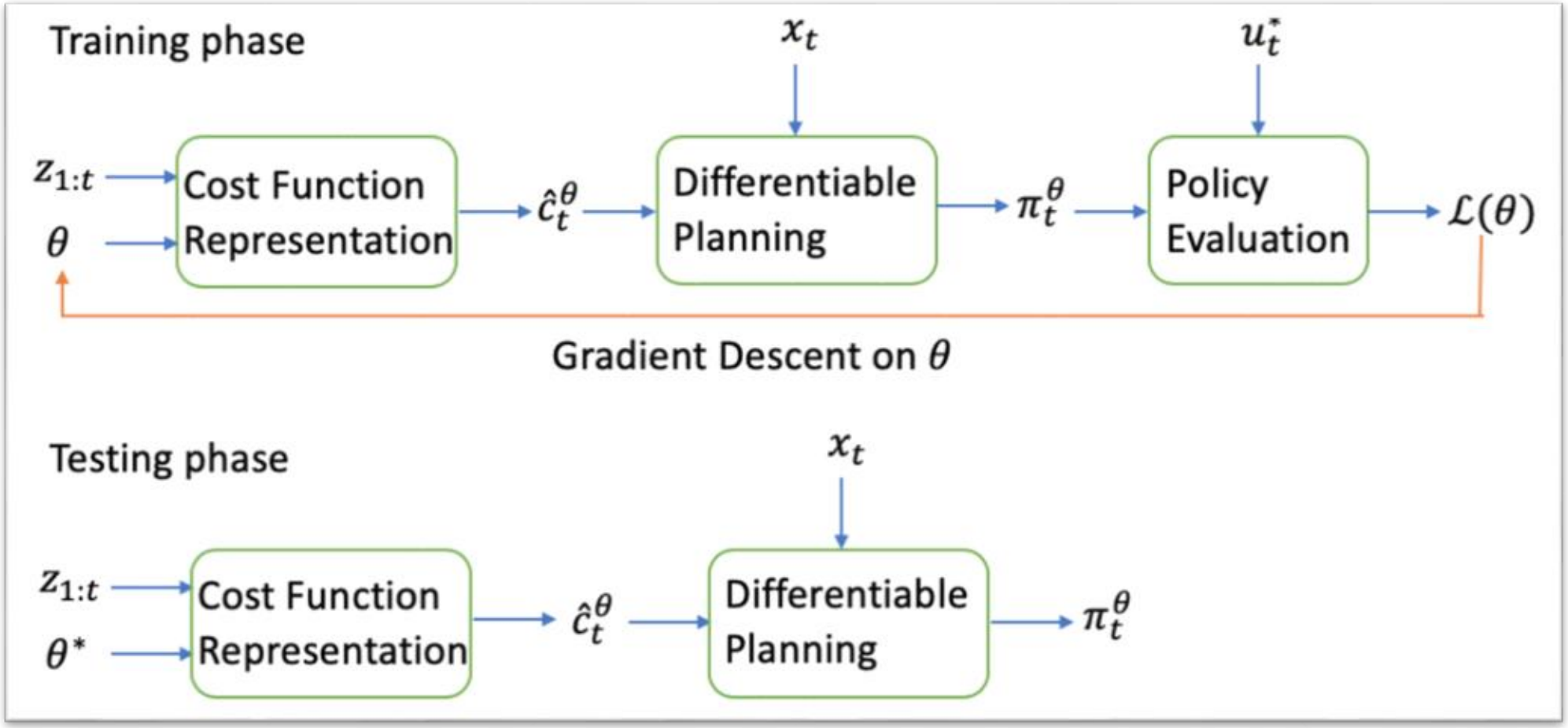
Prior Work: Dyna-Q

- **Method**: Combines direct reinforcement learning with **planning over a learned world model**, updating the model online.
- **Weakness**: No explicit, parameterized cost representation that can be **learned from expert demonstrations** via gradient descent (**not differentiable**)…



Method — Cost Estimation + Differentiable Planning

- Components: robot state $\boldsymbol{x_t} \in X$; noisy local obs $\boldsymbol{z_t} \in \boldsymbol{Z}$; control $\boldsymbol{u_t} \in U$; deterministic dynamics $\boldsymbol{x_{t+1}} = \boldsymbol{f}(\boldsymbol{x_t}, \boldsymbol{u_t})$;
- Training Set: N expert trajectories $\{(\boldsymbol{x_{t,w}}, \boldsymbol{u^*_{t,w}}, \boldsymbol{z_{t,n}})\}$ of length T_n
- Goal: Learn a parameterized $\boldsymbol{\theta}^*$ cost estimate and a stochastic policy $\boldsymbol{\pi_t^\theta}$ from partial noisy demonstrations



Differentiable Planning: Probabilistic Occupancy Map

VAR

STATE SPACE & OBSERVATIONS

The state space is discretized into N cells. The robot observes binary occupancy of K surrounding cells: $z_t \in \{-1, +1\}^K$.

VAR

FIELD OF VIEW

A binary observation matrix $H(x_t) \in \{0,1\}^{K \times N}$ encodes the robot's field of view at state x_t.

VAR

OBSERVATION MODEL

A parameterized observation model $p(z_t \mid x_t, m; \theta)$ — implemented as a sigmoid or neural network — models sensor noise.

VAR

POSTERIOR MAP BELIEF

The posterior map belief $p^\theta_t(m) := p(m \mid z_{1:t})$ is updated via Bayes' rule (log-odds form) as new observations arrive.

VAR

INDUCED COST FUNCTION

The cost function $\hat{c}^\theta_t$ is induced from the occupancy probability of each cell — high occupancy probability → high cost.

➢ **Key Insight:**

The cost representation is **online-updatable** and **differentiable** with respect to θ.

Differentiable Planning: Local Expansions

Algorithm 1 Differentiable A* Planning

Input: Current state x_t, goal state x_g, cost function $\hat{c}_t^\theta$, heuristic function h, transition function f
$g(x_g) = 0,\ g(x) = \infty\ \forall x \in \mathcal{X}$
OPEN $\leftarrow \{x_g\}$, CLOSED $\leftarrow \{\}$
while $x_t \notin$ CLOSED **do**
    Remove x with smallest $g(x) + h(x)$ value from OPEN and insert x in CLOSED
    for $u \in \mathcal{U}$ **do**
        $g(y) \leftarrow \textsc{Conv}(g(s), \delta(f(s,u) - x))$
        $c(y,u) \leftarrow \textsc{Conv}(\hat{c}_t^\theta(s,u), \delta(f(s,u) - x))$
        $g(y) \leftarrow \min\{g(y), c(y,u) + g(x)\}$
    end for
end while
$Q(x_t, u_t) \leftarrow \hat{c}_t^\theta(x_t, u_t) + g(f(x_t, u_t))\ \forall u_t \in \mathcal{U}$
return policy $\pi_t^\theta(u_t|x_t) = \frac{\exp(-Q(x_t,u_t))}{\sum_{u' \in \mathcal{U}} \exp(-Q(x_t,u'))}$

Key 1: access to predecessors is implemented as **convolutional operations** with delta kernels, enabling gradient flow.

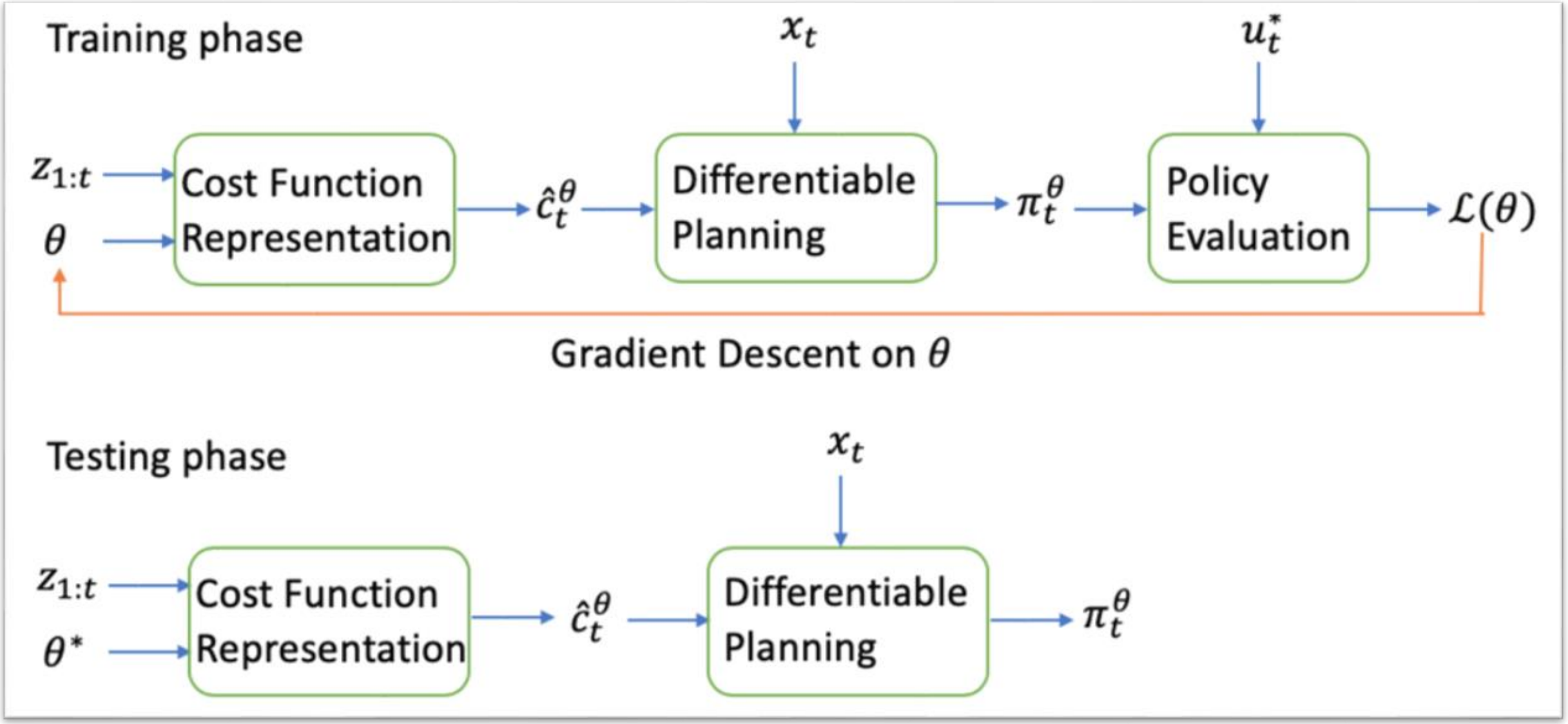
Key 2: **min-pooling** is also a standard differentiable operators in deep learning frameworks, allowing for seamless backpropagation.

Key 3: The **softmax formulation** ensures the policy is differentiable, allowing loss to be backpropagated through the entire planner to the cost function parameters $\boldsymbol{\theta}$.

➢ **Key Insight:**

Expansions occur **only on OPEN list**, not the full grid — **far more efficient** than VIN.

Review Training and Testing Pipeline



Result: **Ours A*** V.S. **VIN** in 2D Grid Navigation

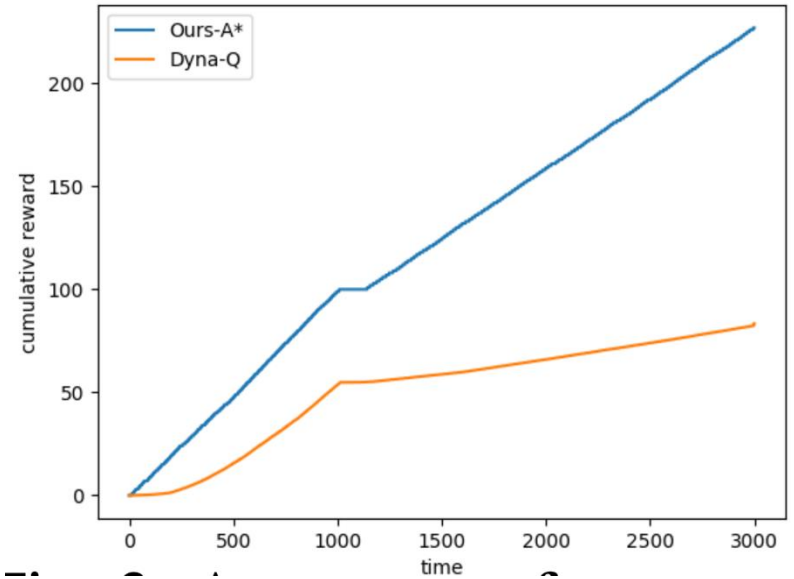
- Setting:
  - 2D grid world with randomly generated obstacles;
  - Fully observable environment;
  - Expert controls computed by Dijkstra's algorithm.
- Metrics: Success Rate ↑; Trajectory Discrepancy ↓

Models	Succ. rate	Traj. diff.
VIN	95.12 %	0.264
Ours-A*	100 %	0.183

- Summary: The differentiable A* planner produces trajectories that **more closely match optimal** expert paths, as its heuristic **guides the planner toward the goal**, focusing **cost learning on regions critical** for successful task completion.

Result: **Ours A*** V.S. **Dyna-Q** in Changing Environments

- Setting:
  - Changing environment where the optimal path switches mid-episode;
  - Robot receives only partial observation of the next state;
- Metrics: Cumulative Reward Curves



- Summary: The **online-updatable** occupancy cost representation is critical for rapid adaptation, where the model **does not need to re-train from scratch when the environment changes**.

End-to-end differentiable planning enables scalable, adaptive cost learning from demonstrations

- Contributions:
  - Probabilistic occupancy-based cost representation — online updatable from partial, noisy observations
  - Differentiable A* planning — efficient local expansions via convolution and min-pooling
  - End-to-end trainable model — backpropagates policy loss through the planner into the cost function parameters
- Results:
  - Outperforms VIN (100% vs. 95.12% success)
  - Faster adaptation in changing environments compared to Dyna-Q

Future Work

- Limitations:

Evaluated on 2D discrete grid; scalability to continuous, high-dimensional state spaces remains open.

- Future Directions:
  - Extension to 3D environments and continuous state/action spaces
  - Integration with visual observation models (e.g., raw RGB images)
  - Application to multi-robot cooperative navigation