LEARNING TO SEARCH FROM DEMONSTRATION SEQUENCES

Dixant Mittal, Liwei Kang, Wee Sun Lee

INTRODUCTION

Search and Planning

are fundamental for complex reasoning tasks

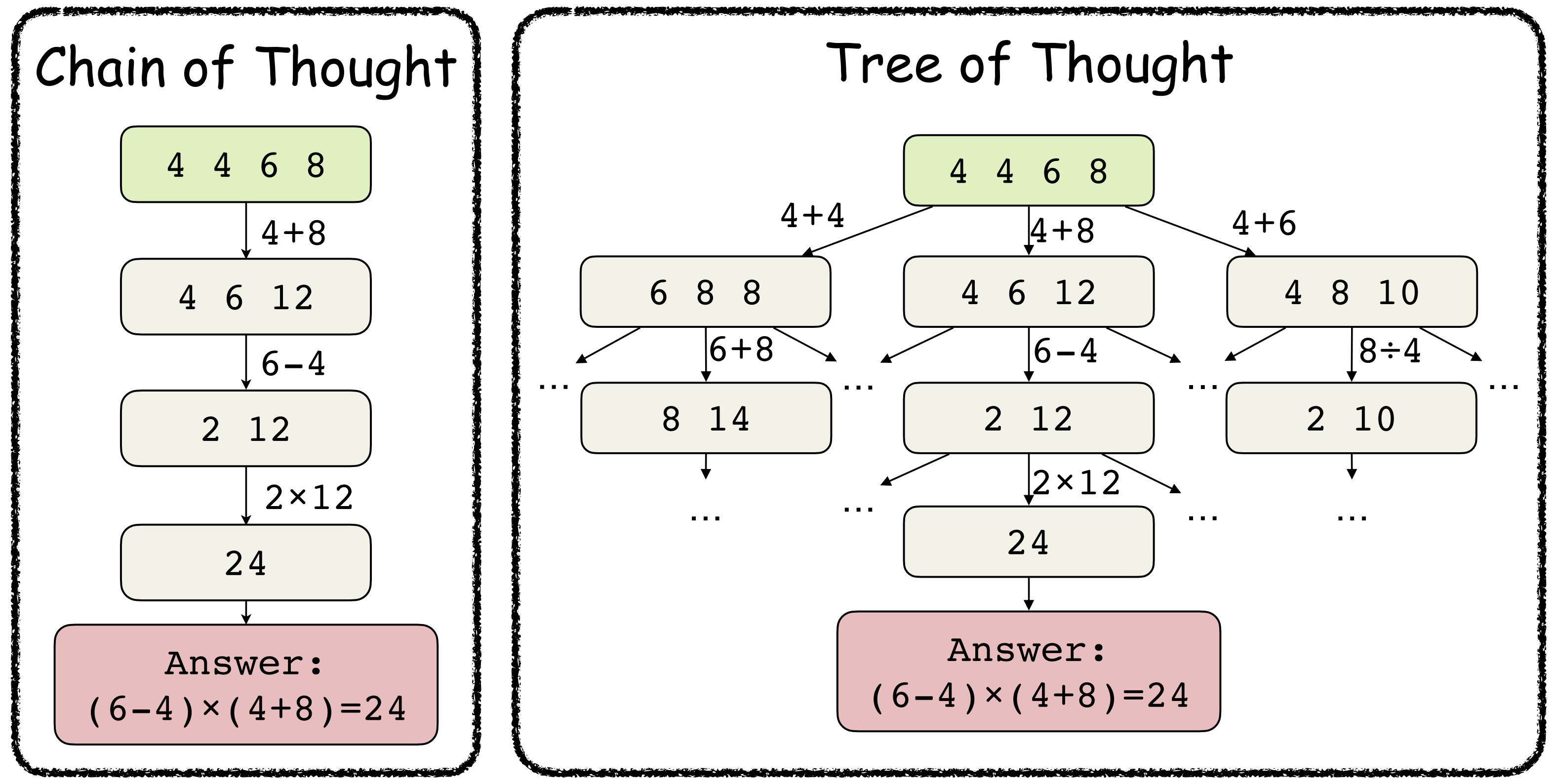
Data is Often Limited

in real-world and all we may have is a collection of

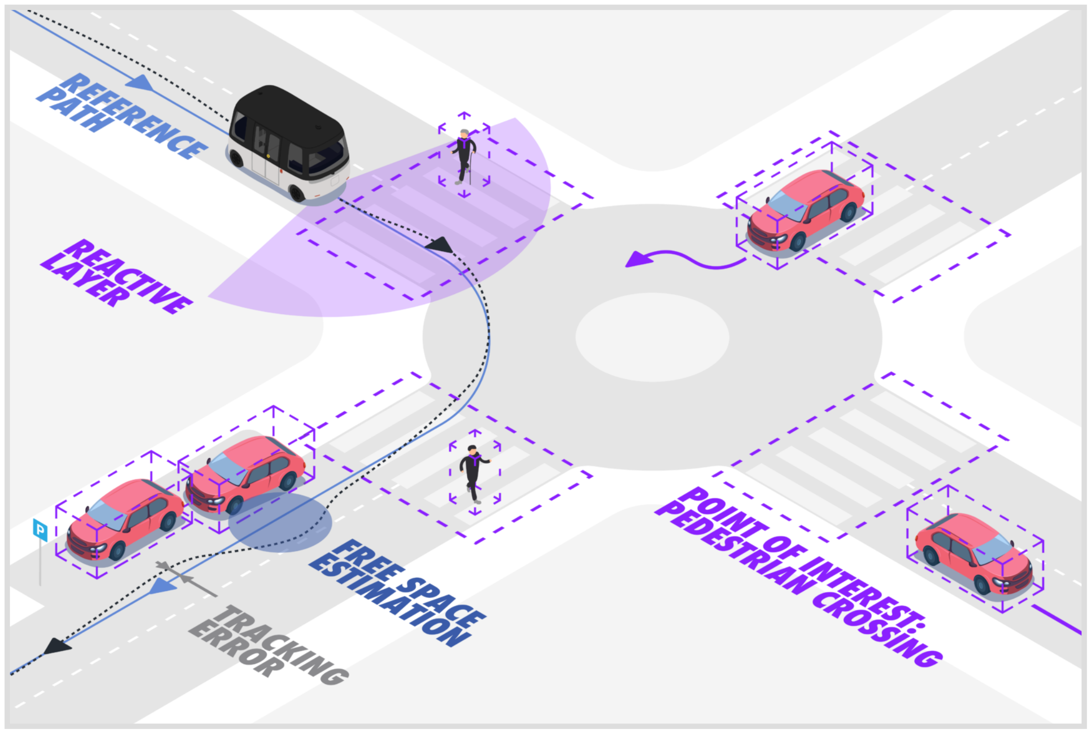
Demonstration Sequences

Learning to Search from Demonstrations is tricky because:

- Limited **state space coverage**
- **No exploration**
- **Compounding Errors** for planning



Reasoning in LLMs

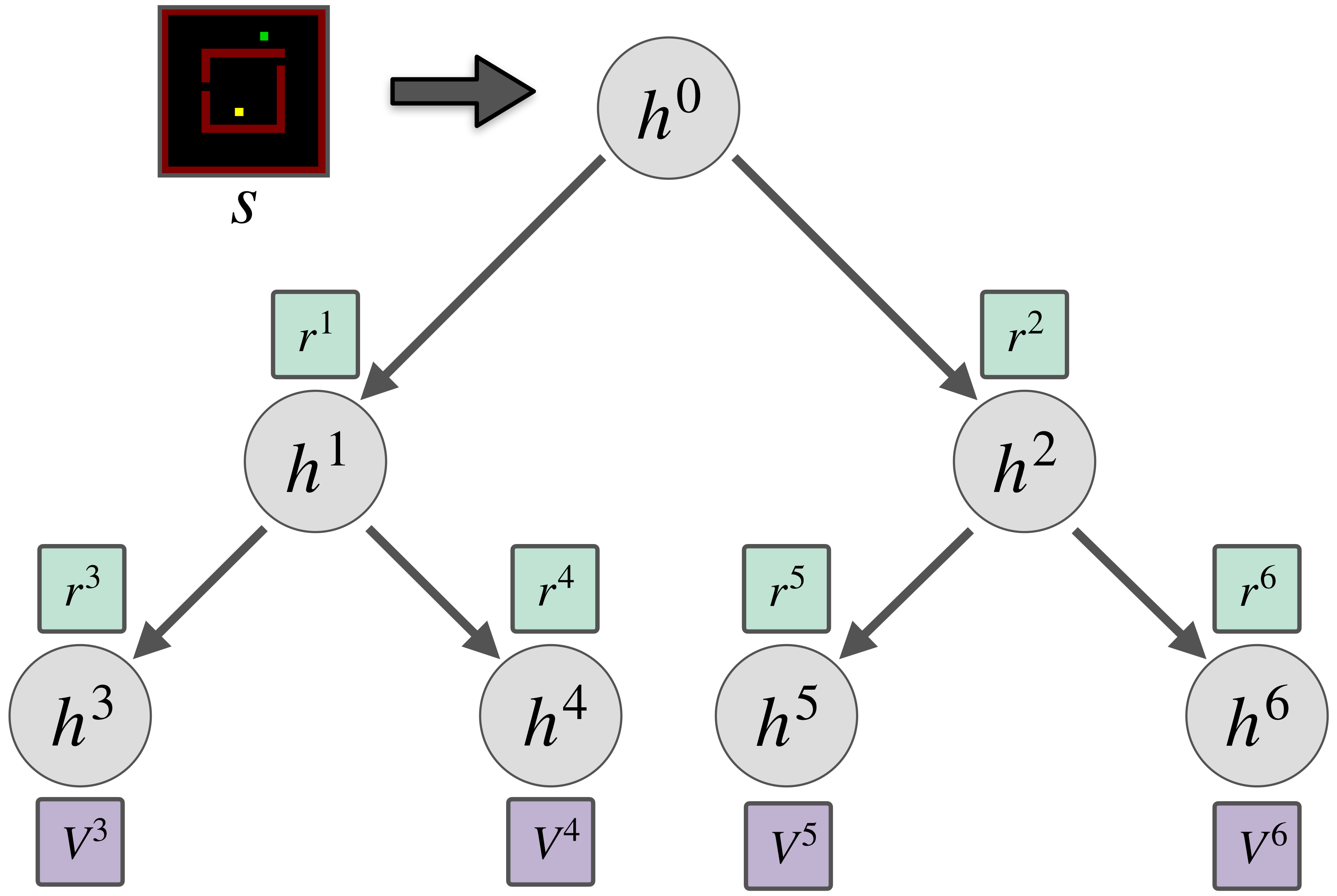


Real World
Autonomous Driving

EXISTING APPROACHES

Search Tree as Parametric Policy
and directly learn the mapping from state to action

TreeQN
expands the full search tree upto a fixed depth
and backups the value to the root node



TreeQN

TREEQN: **STRENGTHS & LIMITATIONS**

STRENGTHS

Planning inductive bias

No dependency on Simulator

LIMITATIONS

TreeQN is **exponential** in depth

Infeasible to perform **deeper** search

Performance suffers in **complex** problems

DIFFERENTIABLE TREE SEARCH NETWORK (D-TSN)

Modular Neural Network Architecture

that comprises of several *learnable submodules*

Algorithmic Inductive Bias

of a flexible and scalable *best-first tree search* algorithm

Learnt World Model

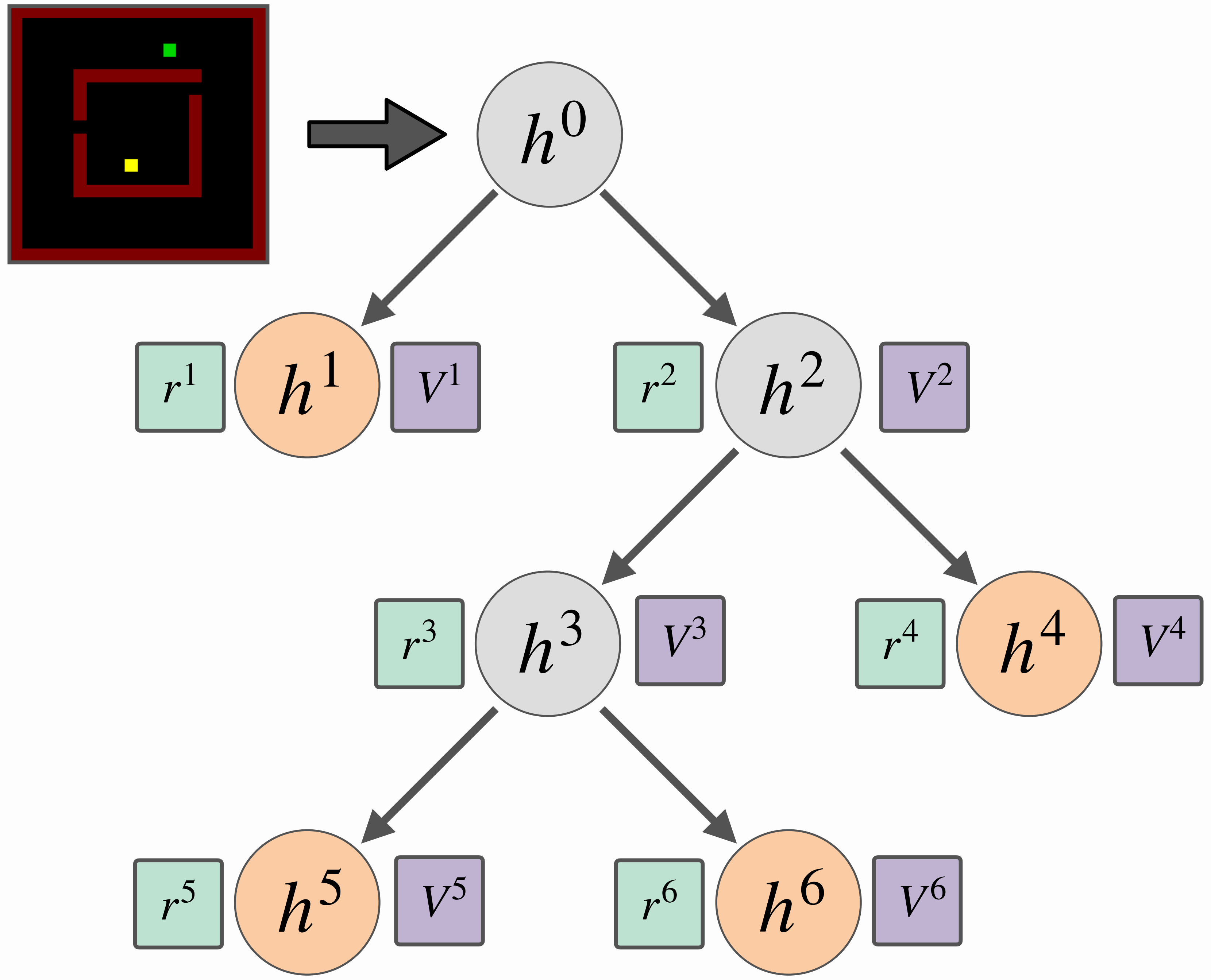
that is trained to be *useful* for the online search, even if inaccurate

Joint Optimisation

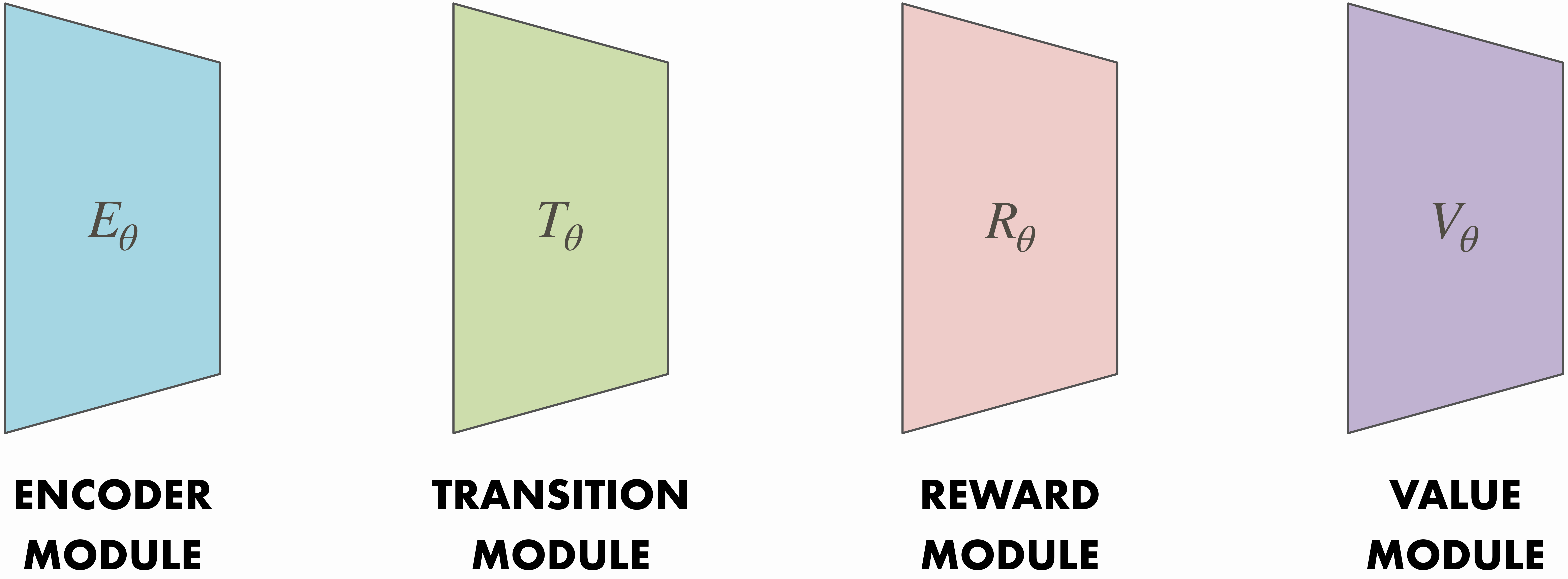
Trains the *search* and *world model* submodules jointly

Additional Technical Details

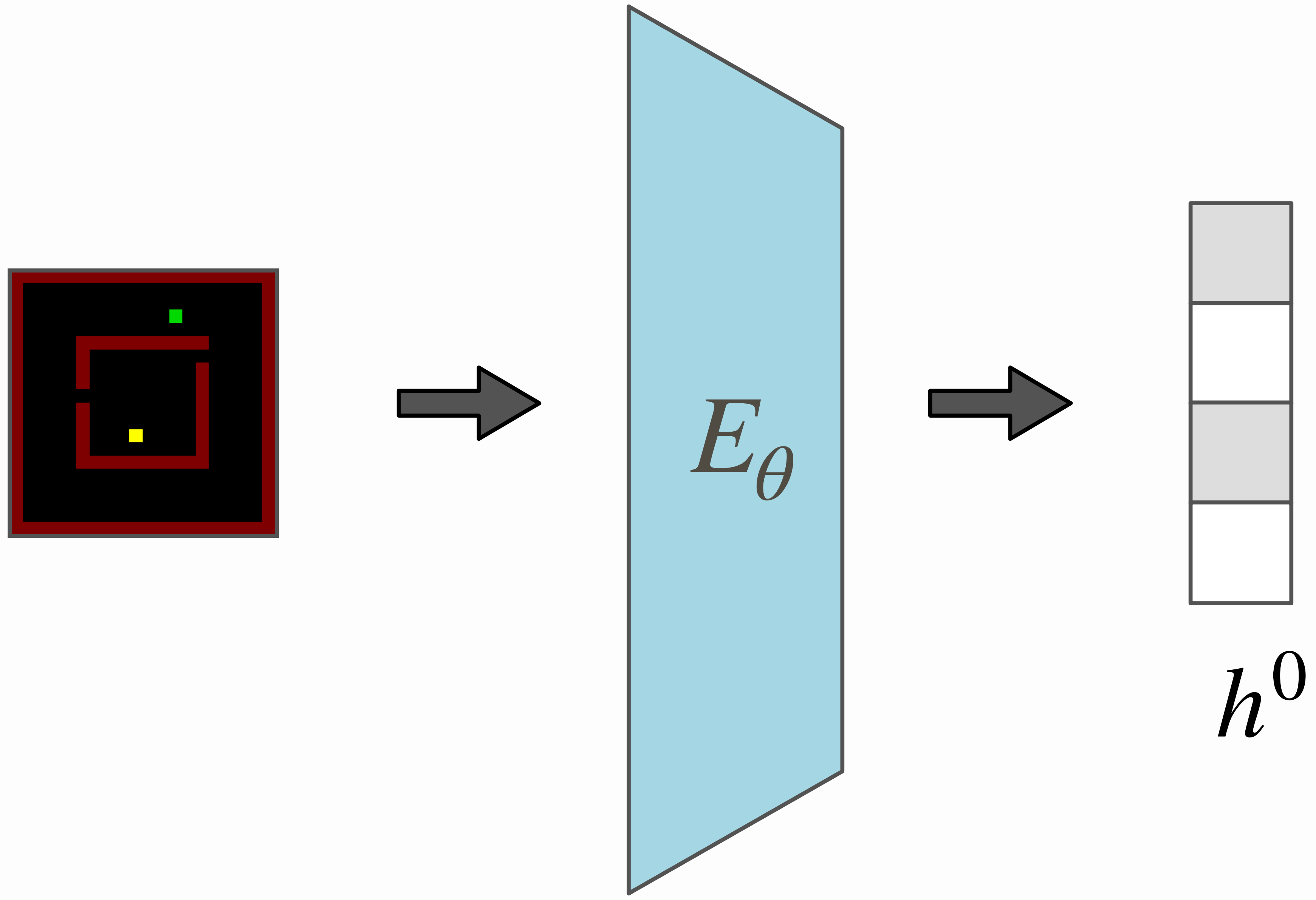
like *Tree Expansion Policy* and *Telescopic Sum for Variance Reduction*

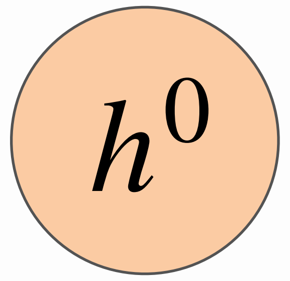


D-TSN: **SUBMODULES**

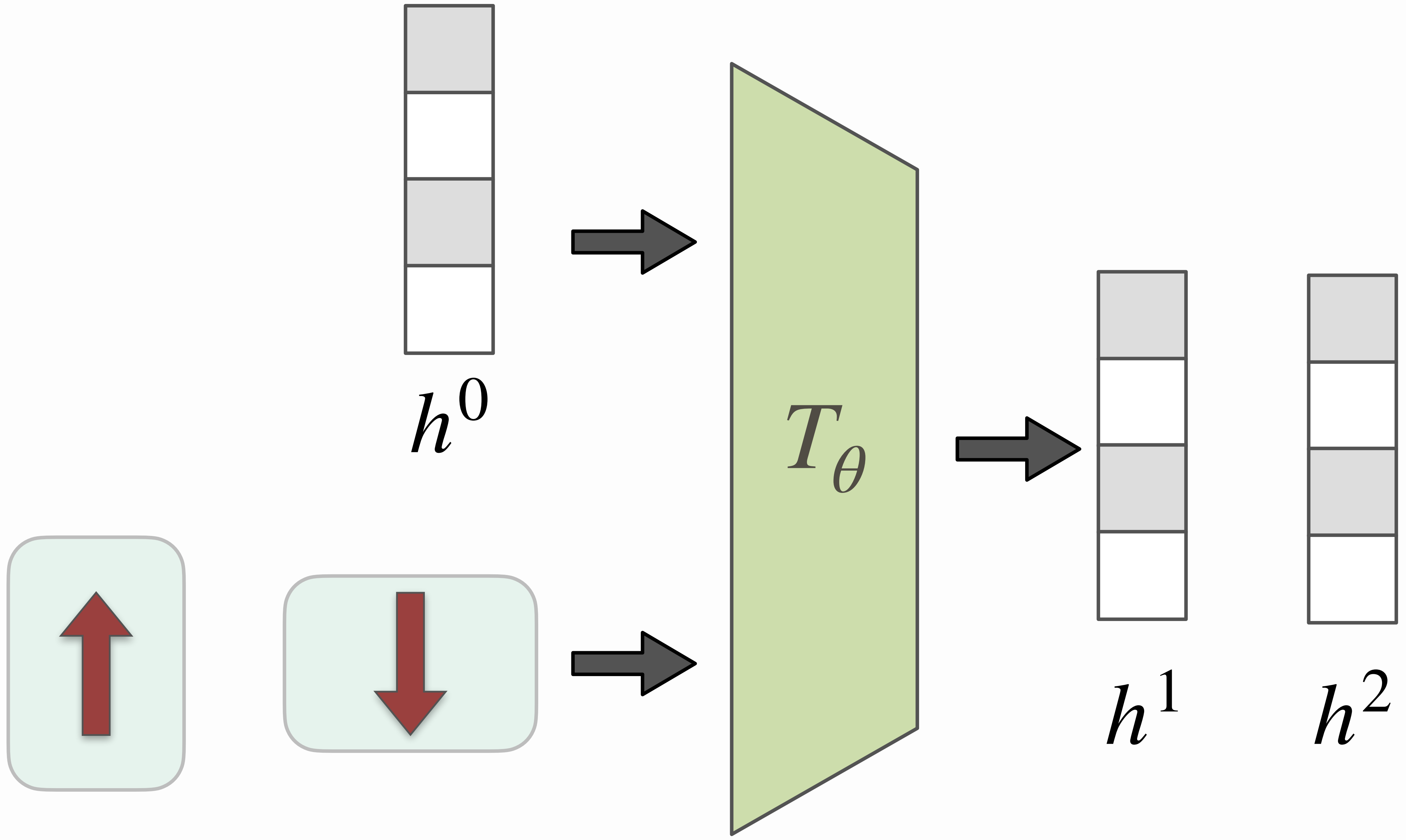


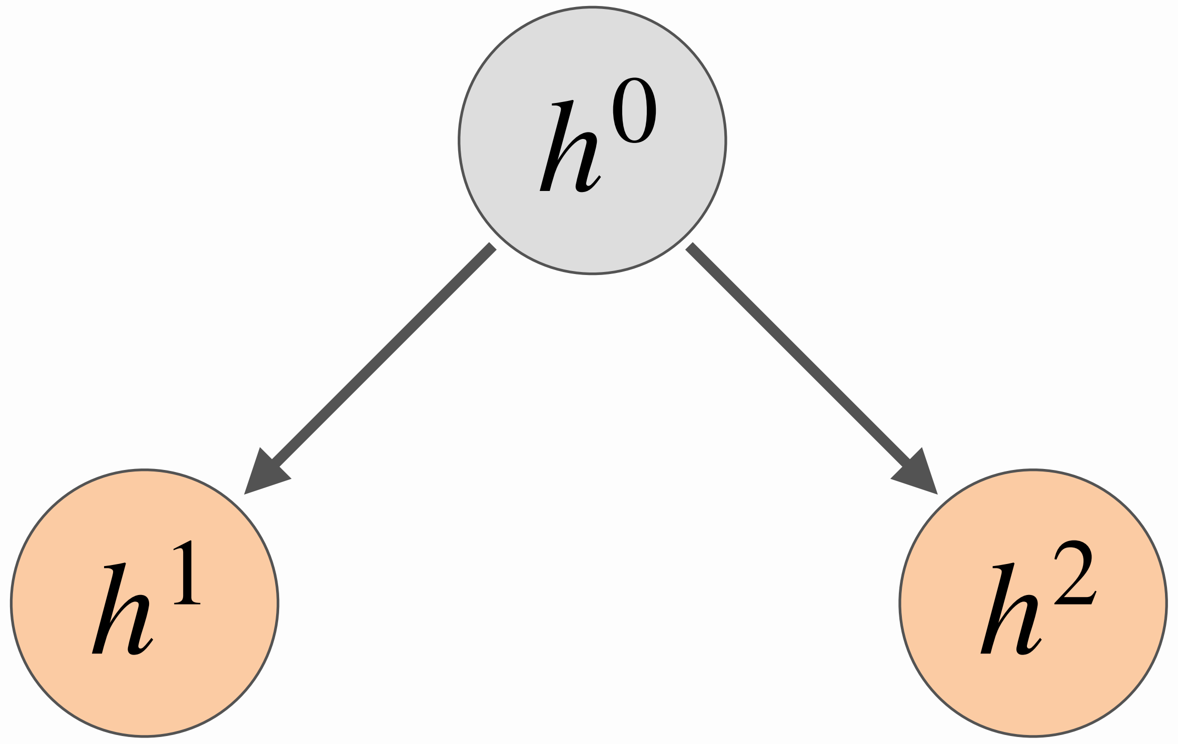
D-TSN: **ENCODING**



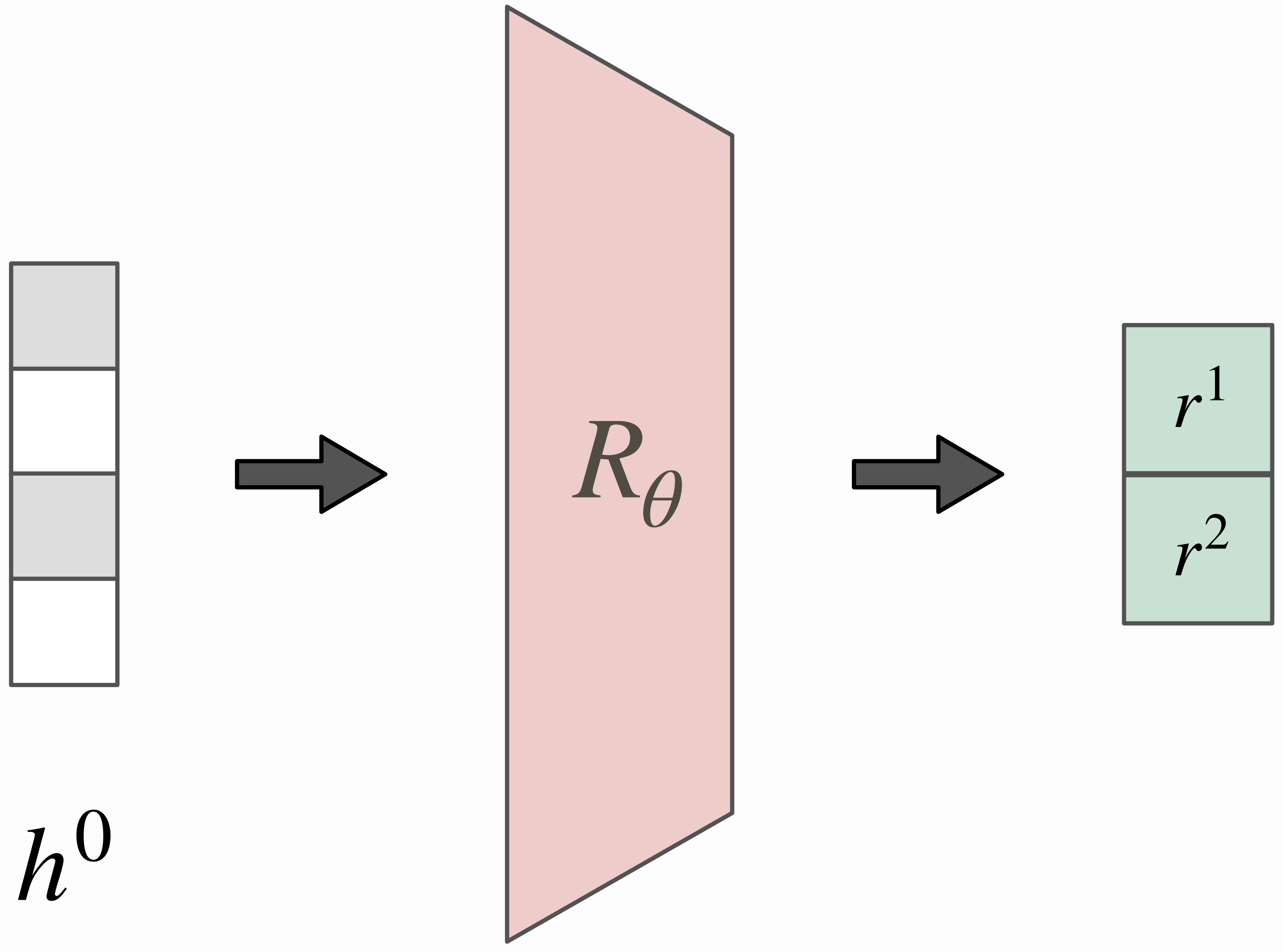


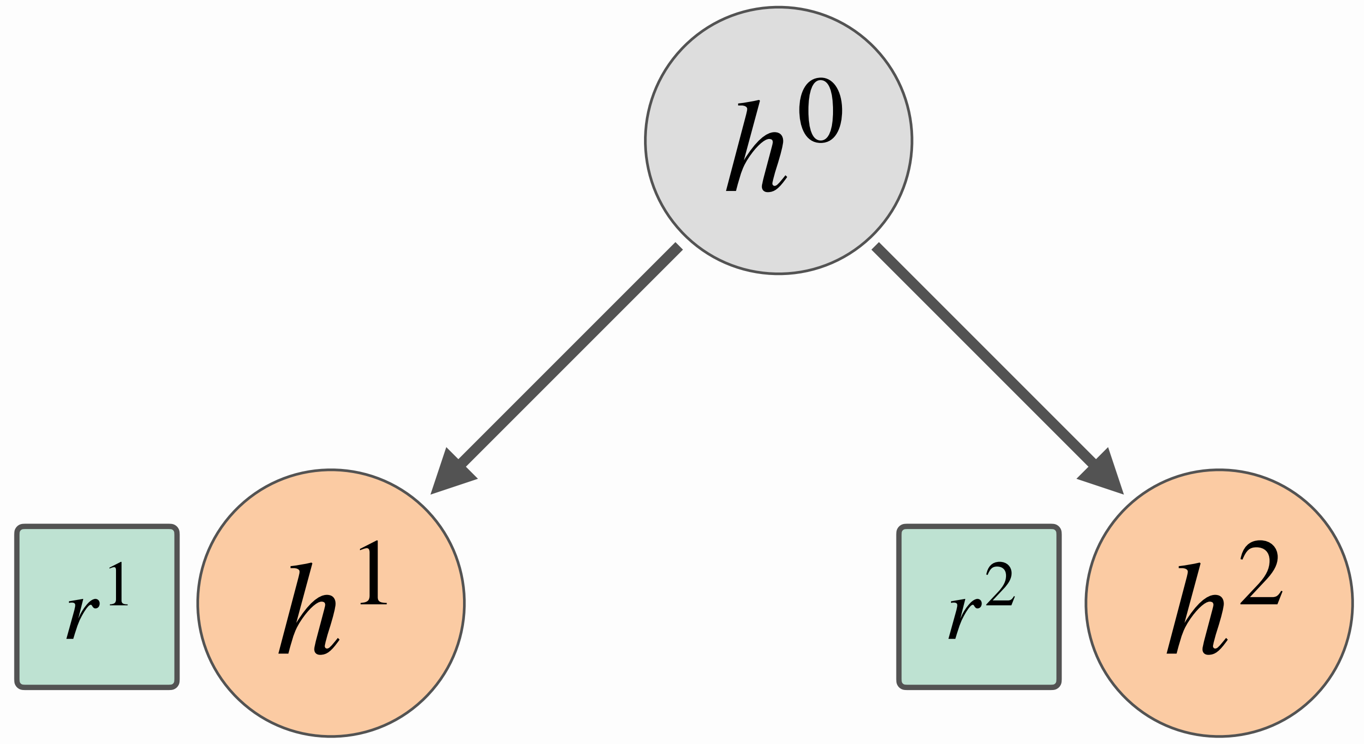
D-TSN: EXPANSION PHASE



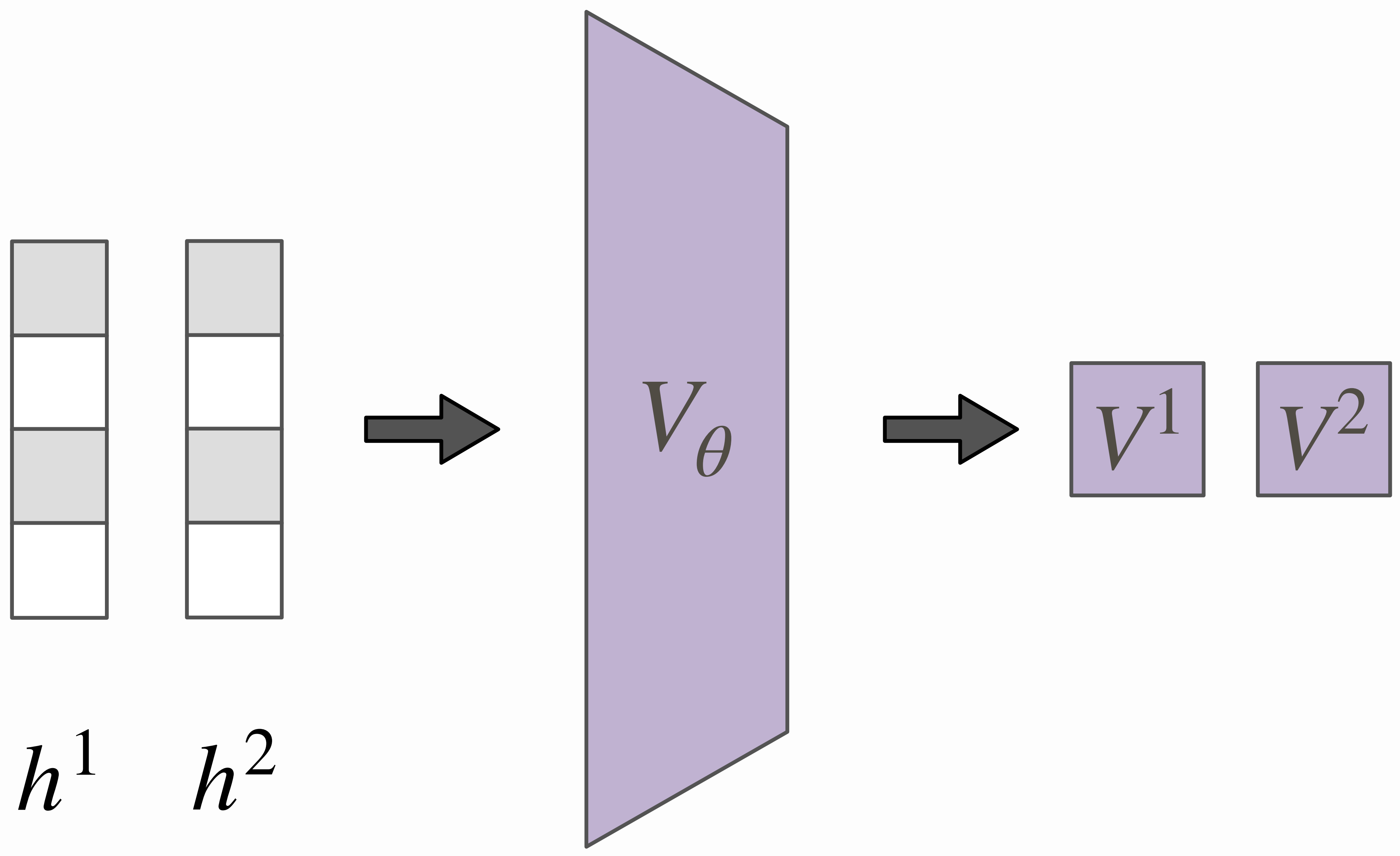


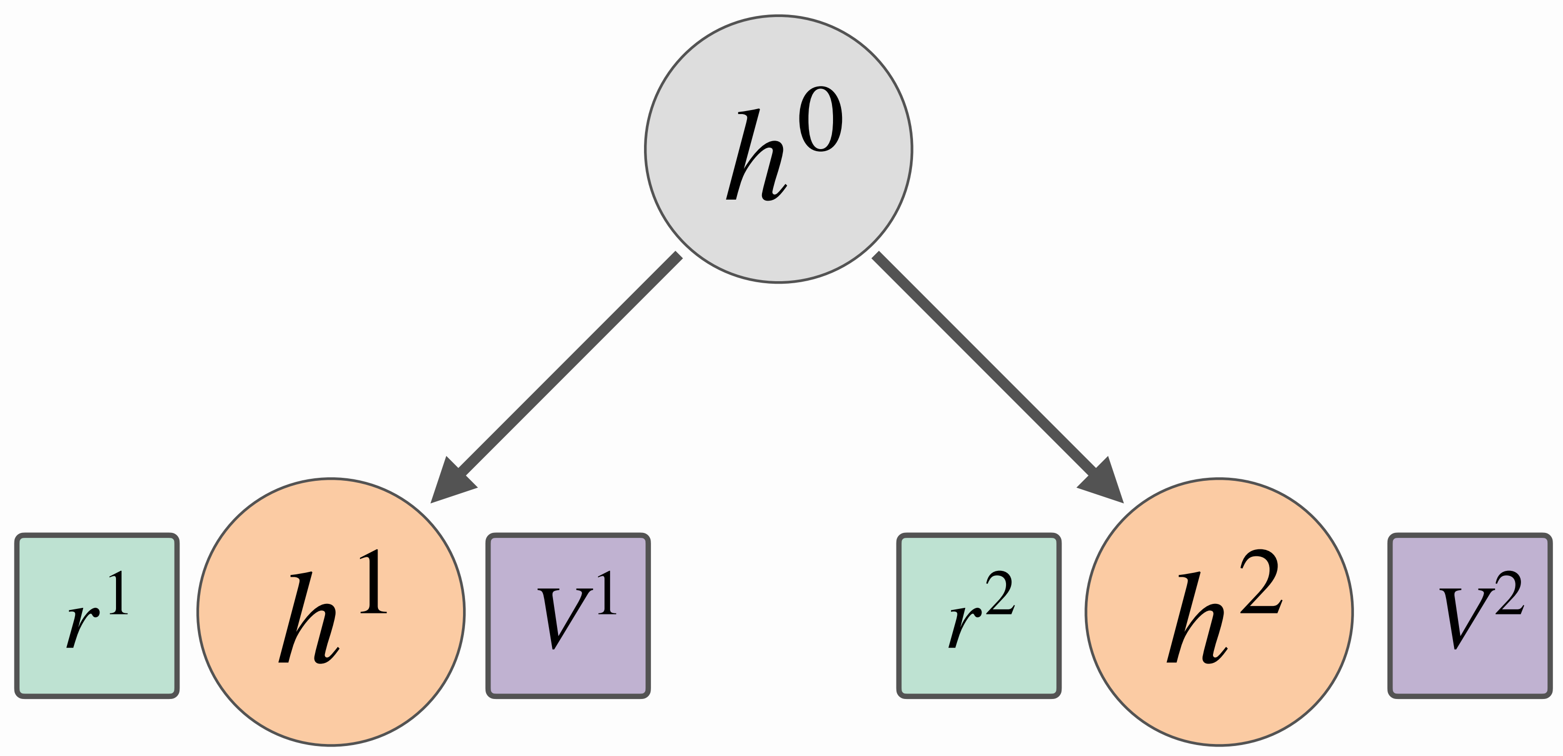
D-TSN: EXPANSION PHASE





D-TSN: **EXPANSION PHASE**

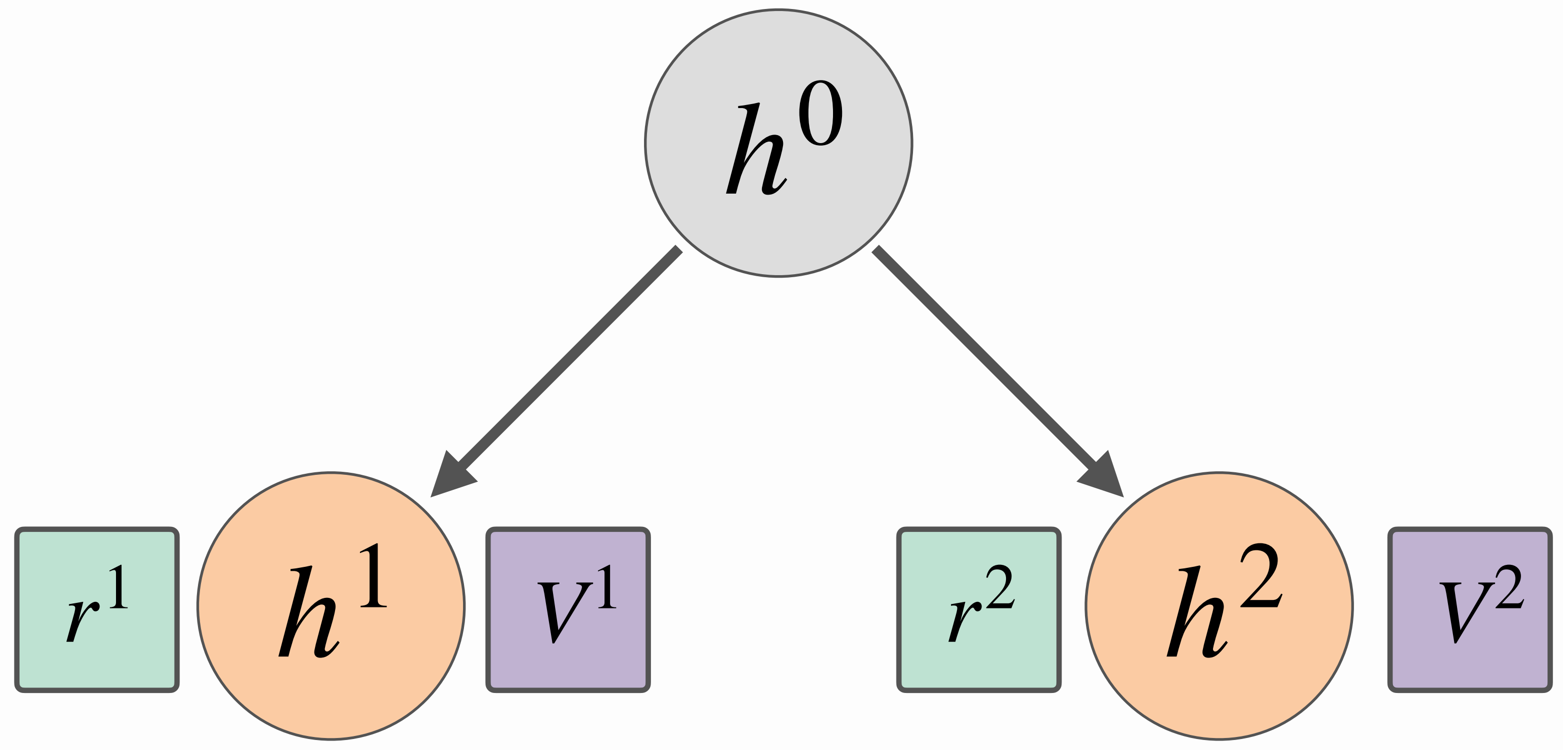




D-TSN: **EXPANSION PHASE**

Stochastic Tree Expansion Policy

$$n \sim \pi(\tau)$$



τ_1

$$\pi(\tau_1) = \textbf{softmax}\begin{pmatrix} h^1 & h^2 \end{pmatrix}$$

$$= \textbf{softmax}\begin{pmatrix} r^1 + V^1 \\ r^2 + V^2 \end{pmatrix}$$

NUS National University of Singapore

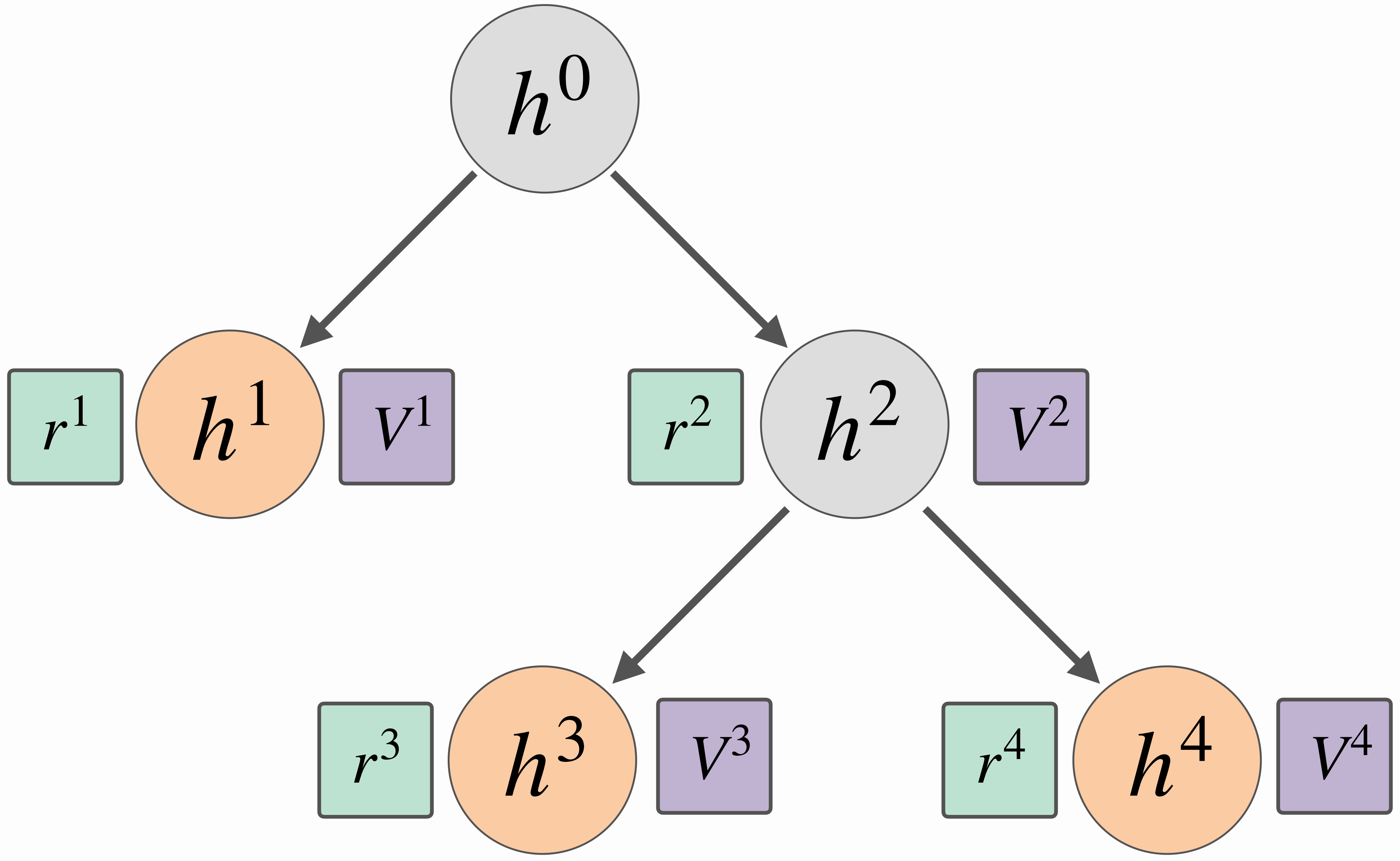
D-TSN: EXPANSION PHASE

Stochastic Tree Expansion Policy

$$n \sim \pi(\tau)$$

$$\pi(\tau_2) = \textbf{softmax}\left(h^1 \quad h^3 \quad h^4 \right)$$

$$= \textbf{softmax}\begin{pmatrix} r^1 + V^1 \\ r^2 + r^3 + V^3 \\ r^2 + r^4 + V^4 \end{pmatrix}$$



τ_2

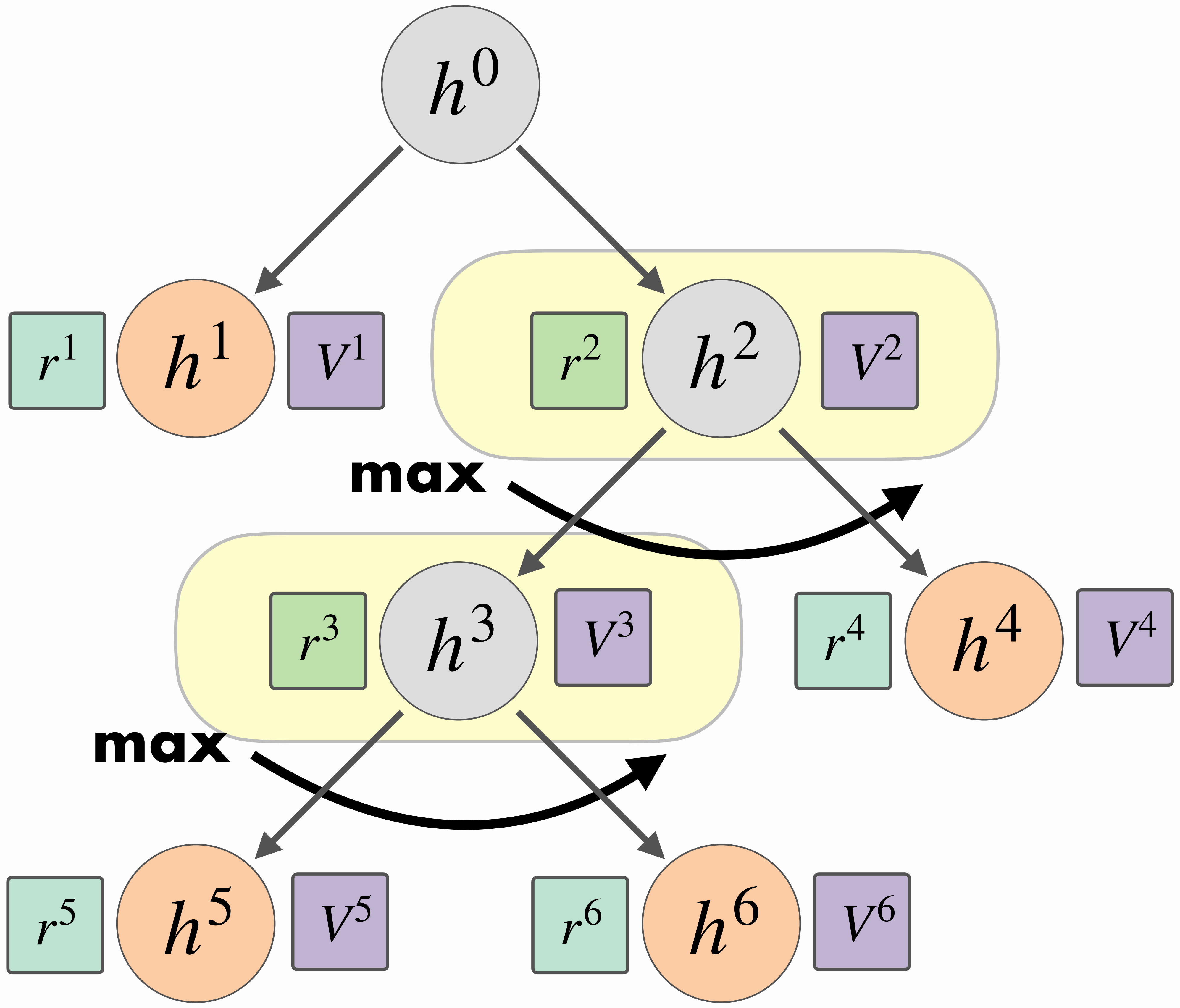
NUS National University of Singapore

D-TSN: BACKUP PHASE

BELLMAN EQUATION

$$Q(h, a) = r(h, a) + V(h')$$

$$V(h) = \max_{a} \left[Q(h, a) \right]$$



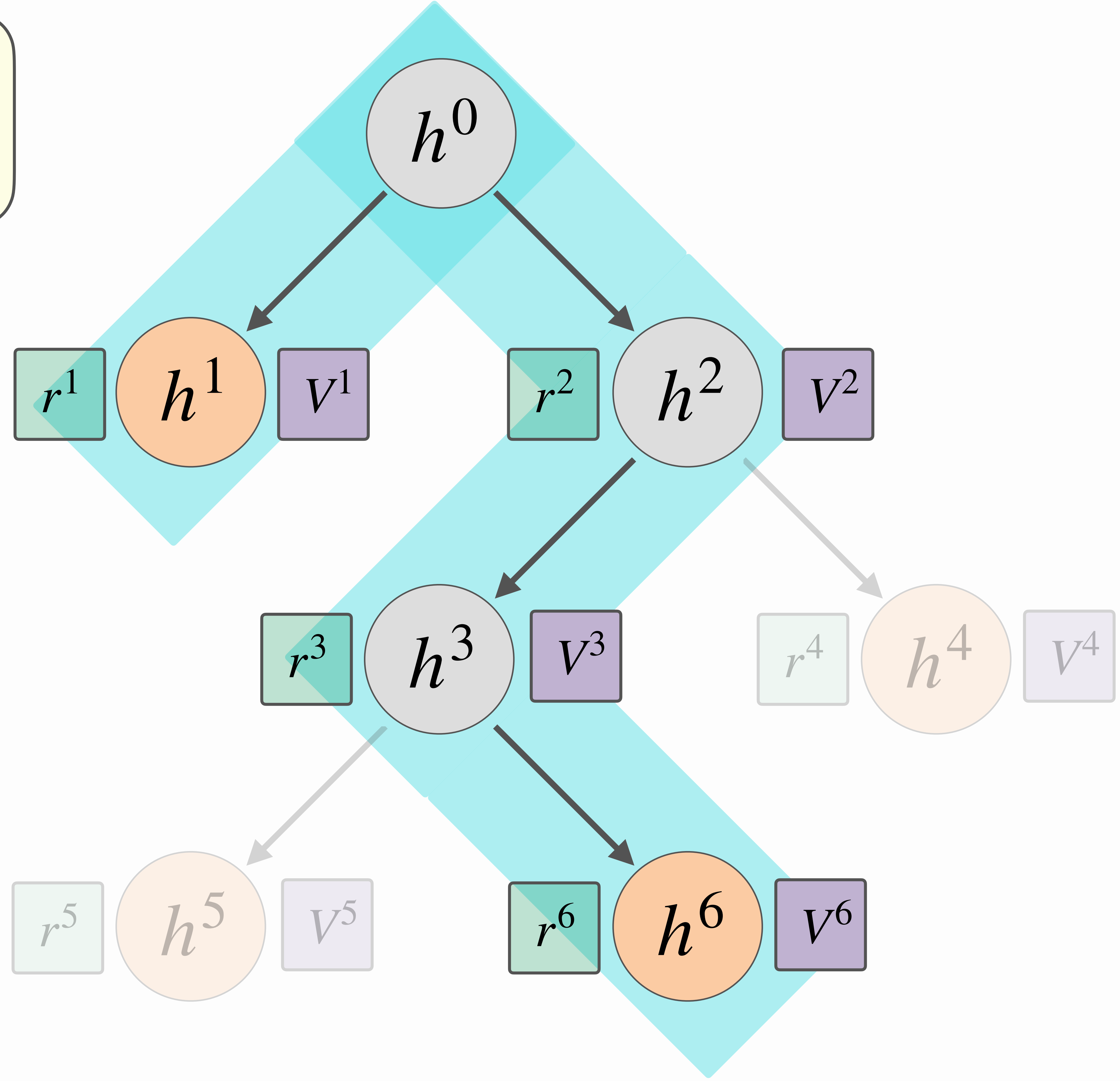
D-TSN: **OUTPUT**

Model's Output: $Q_\theta(s,\ a)$

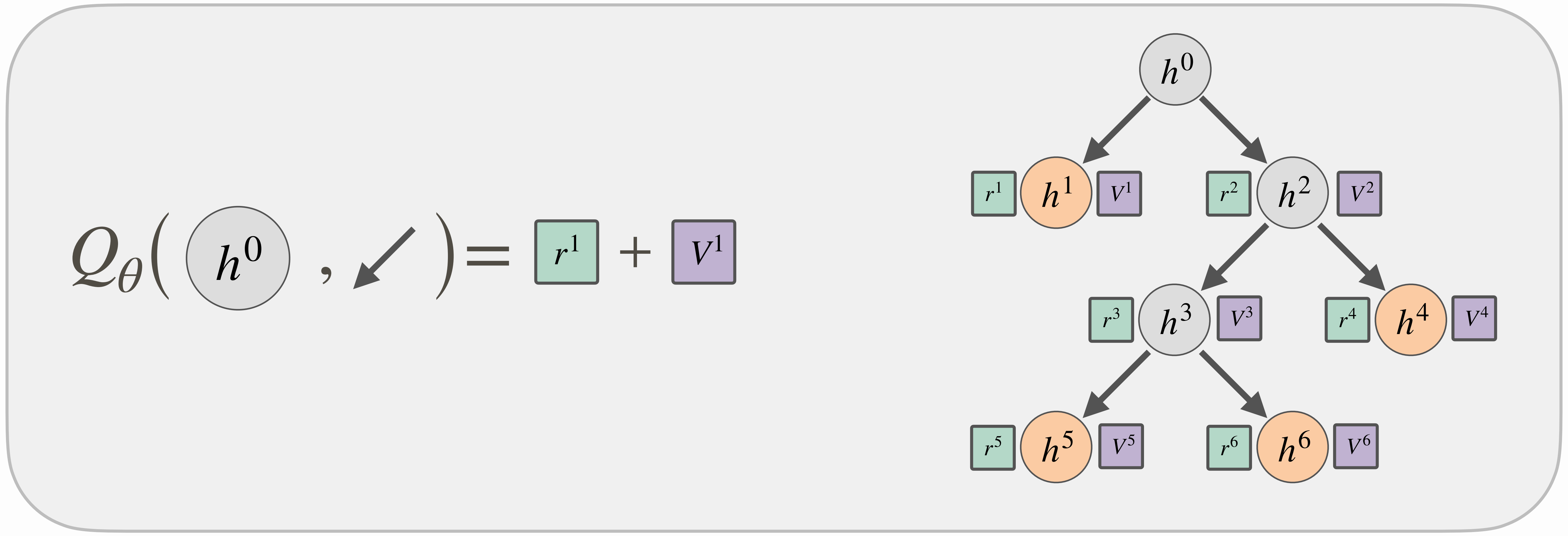
Path returning the highest value

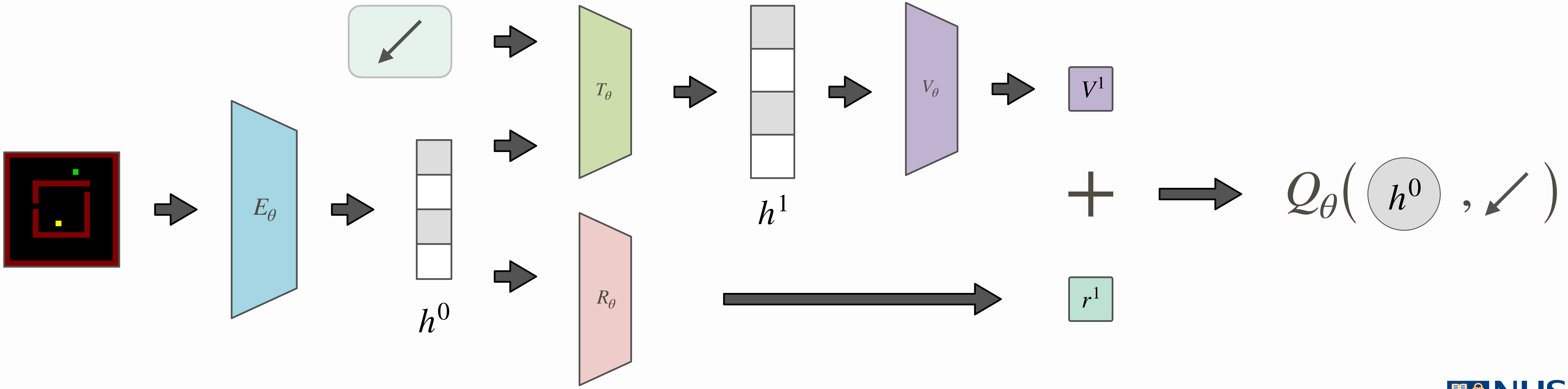
$$Q_\theta(h^0, \swarrow) = r^1 + V^1$$

$$Q_\theta(h^0, \searrow) = r^2 + r^3 + r^6 + V^6$$

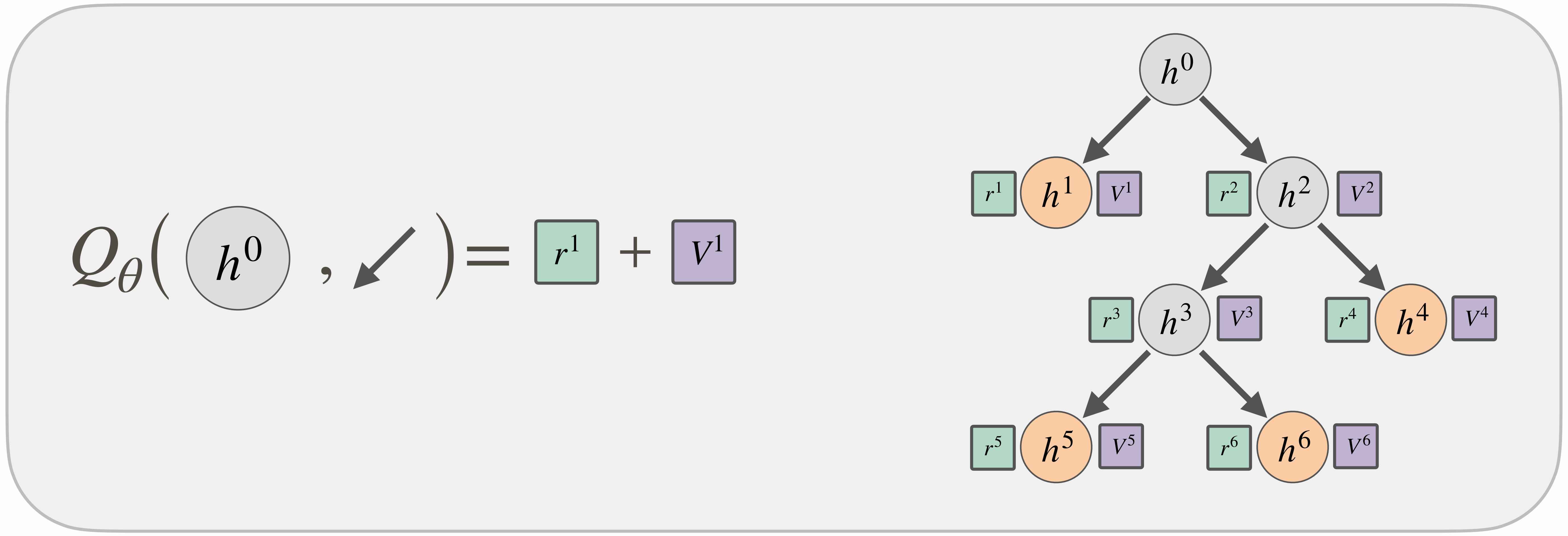


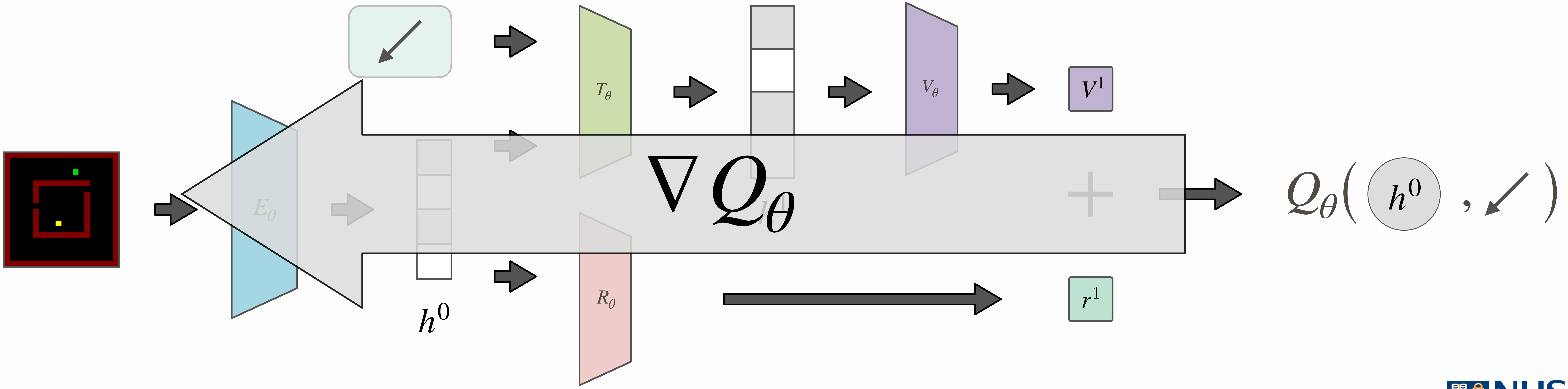
D-TSN: COMPUTATION GRAPH





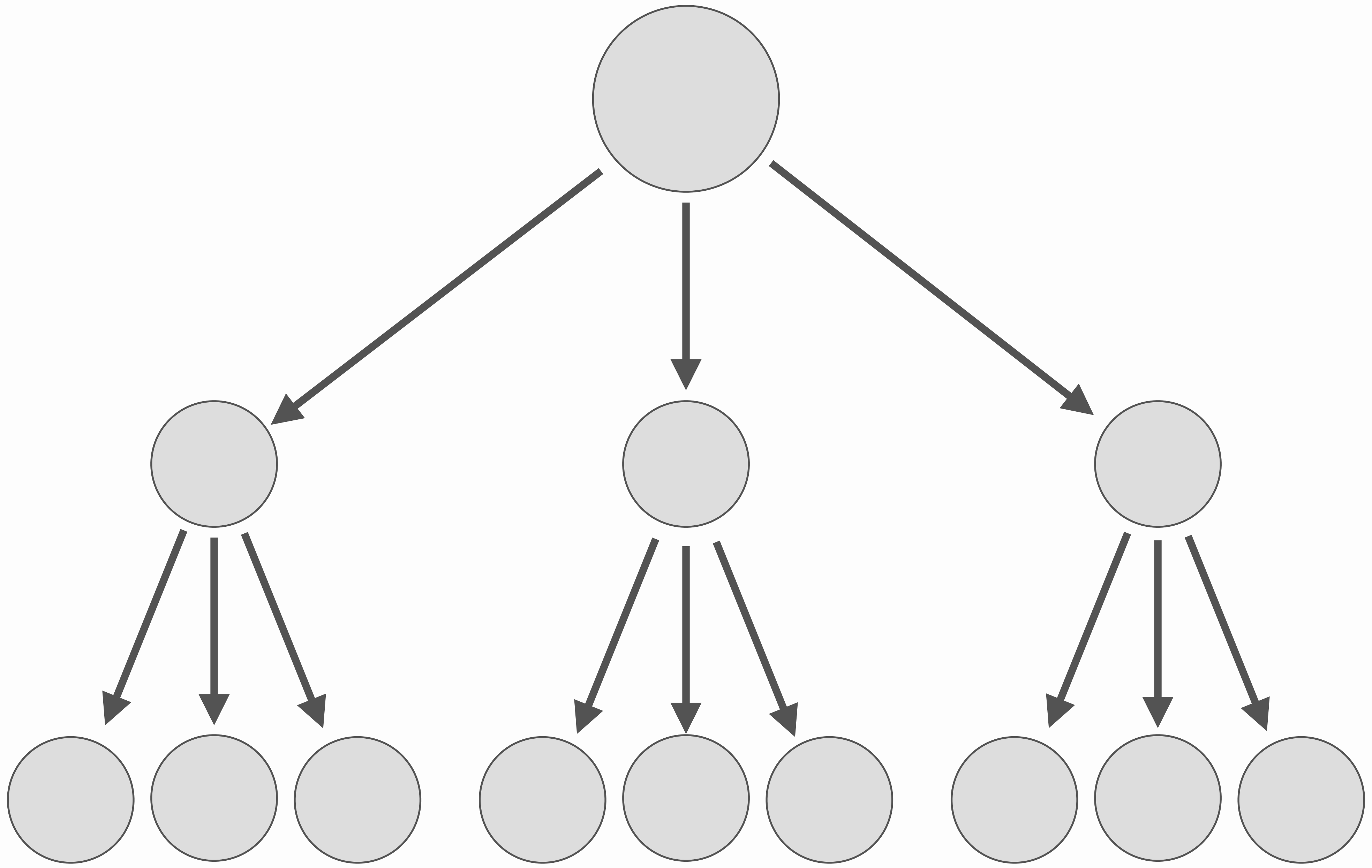
D-TSN: COMPUTATION GRAPH





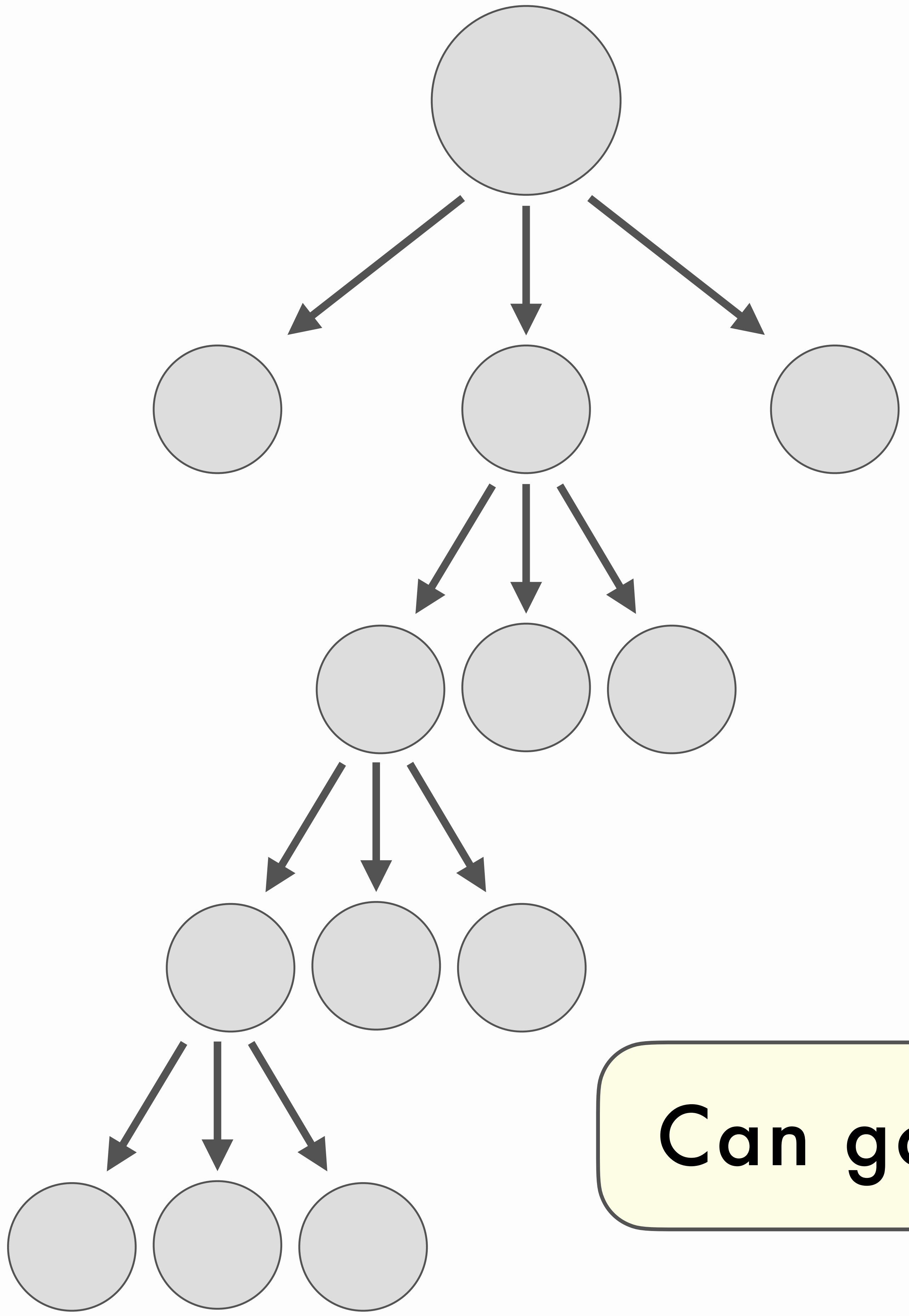
D-TSN: COMPARISON WITH TREEQN

TreeQN



Exponential in Depth

D-TSN



Can go **Deeper**

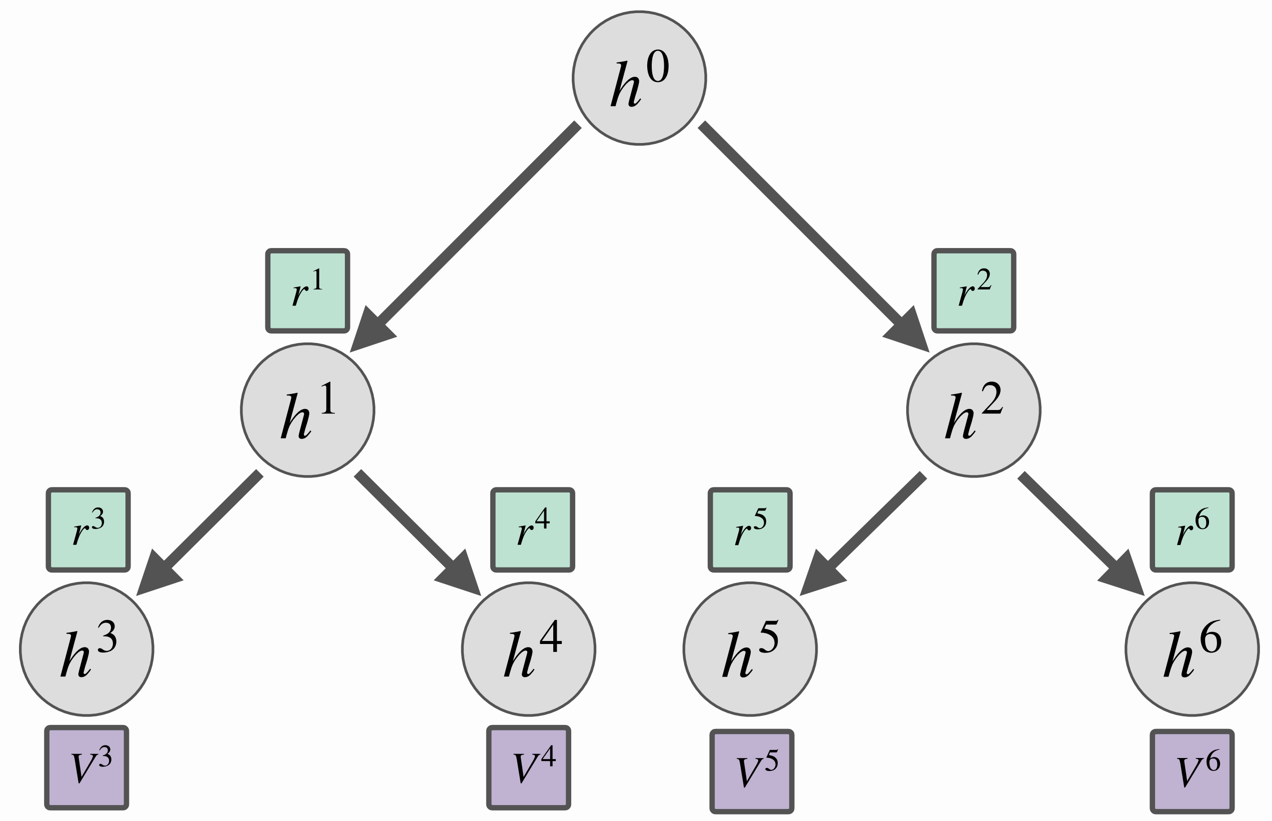
NUS National University of Singapore

D-TSN: CONTINUITY OF LOSS FUNCTION

TreeQN

$$Q(s, Left) = r^1 + max \begin{bmatrix} r^3 + V^3 \\ r^4 + V^4 \end{bmatrix}$$

$$Q(s, Right) = r^2 + max \begin{bmatrix} r^5 + V^5 \\ r^6 + V^6 \end{bmatrix}$$



$Q_\theta(s, a)$ from a **Full Tree Search** are **continuous** in parameter space

$L\Big(Q_\theta(s, a)\Big)$ is **continuous** in parameter space θ

D-TSN: CONTINUITY OF LOSS FUNCTION

D-TSN

A **best-first search** solution is an **approximation** to the **Full Tree Search** solution

$Q_\theta(s, a)$ depends upon the sampled tree τ

$L\left(Q_\theta(s, a \mid \tau)\right)$ can be **discontinuous** in parameter space θ

D-TSN: **EXPECTED LOSS FUNCTION**

$$\textbf{Loss} = \mathbb{E}_\tau \left[L\Big(Q_\theta(s, a \mid \tau) \Big) \right] = \sum_\tau \pi_\theta(\tau)\, L\Big(Q_\theta(s, a \mid \tau) \Big)$$

Continuous

in the parameter space

D-TSN: **EXPECTED LOSS FUNCTION**

$$\textbf{Loss} = \mathbb{E}_\tau \left[L\Big(Q_\theta(s, a \mid \tau) \Big) \right]$$

$$\nabla_\theta(\textbf{Loss}) = \mathbb{E}_\tau \left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(n_t \mid \tau_t) \; L\Big(Q_\theta(s, a \mid \tau_T) \Big) \; + \; \nabla_\theta L\Big(Q_\theta(s, a \mid \tau_T) \Big) \right]$$

Policy Gradient
(to improve tree expansion policy)

Regular Loss Gradient
(on output Q-values)

D-TSN: **VARIANCE REDUCTION**

$$\nabla_\theta(\textbf{Loss}) = \mathbb{E}_\tau\left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(n_t \mid \tau_t)\, L\Big(Q_\theta(s, a \mid \tau_T)\Big) \;+\; \nabla_\theta L\Big(Q_\theta(s, a \mid \tau_T)\Big)\right]$$

High Variance!

D-TSN: VARIANCE REDUCTION

$$\nabla_\theta(\textbf{Loss}) = \mathbb{E}_\tau\left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(n_t \mid \tau_t)\, L\Big(Q_\theta(s, a \mid \tau_T)\Big) \;+\; \nabla_\theta L\Big(Q_\theta(s, a \mid \tau_T)\Big)\right]$$

Let $L_T = L\Big(Q_\theta(s, a \mid \tau_T)\Big)$ **and** $L_0 = 0$

$$L_T = L_T - L_0$$

$$= L_T - L_{T-1} + L_{T-1} - L_0$$

$$= \sum_{t=1}^{T} L_t - L_{t-1}$$

Telescopic Sum

D-TSN: **VARIANCE REDUCTION**

Treat **Tree Expansion** as another decision making problem with goal of reducing Loss value after t^{th} expansion.

Define

- Reward as $r_t = L_t - L_{t-1}$
- Sum of Rewards as $R_t = \sum_{t}^{T} r_t = L_T - L_{t-1}$

D-TSN: VARIANCE REDUCTION

$$\nabla_\theta(\textbf{Loss}) = \mathbb{E}_\tau\left[\sum_{t=1}^{T}\left[\nabla_\theta \log \pi_\theta(n_t \mid \tau_t)\right] L_T \quad + \quad \nabla_\theta L_T\right]$$

$$= \mathbb{E}_\tau\left[\sum_{t=1}^{T}\left[\nabla_\theta \log \pi_\theta(n_t \mid \tau_t)\right]\left[L_T - L_{t-1}\right] \quad + \quad \nabla_\theta L_T\right]$$

Lower variance!

EXPERIMENTS: **BASELINES**

Differentiable Tree Search Network

Differentiable Search + Joint Optimisation

vs

Model-free Q-network

A generic Neural Network based Q-function

Model-based Search

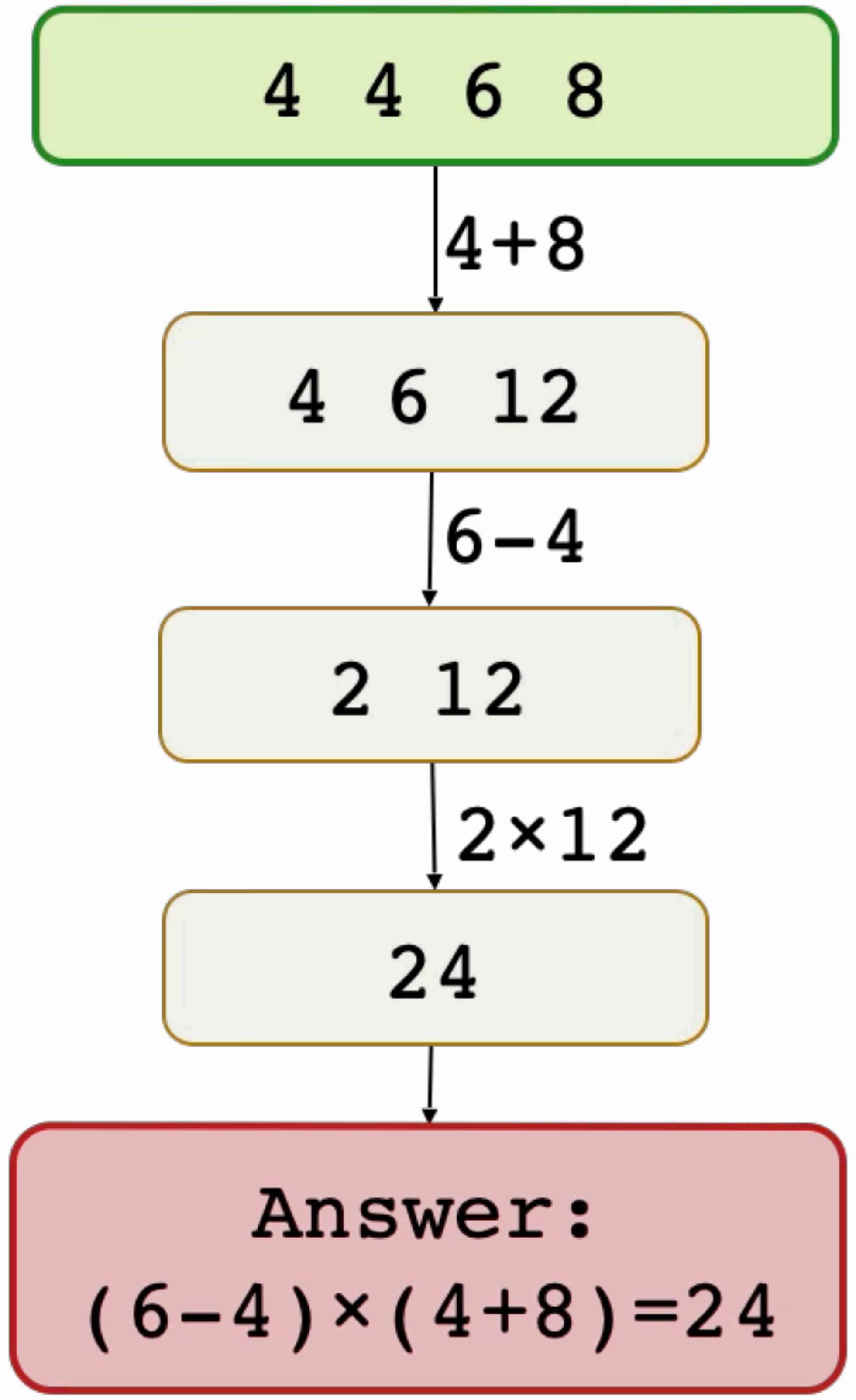
Best-first Tree Search + Learnt World Model
(No joint optimisation)

TreeQN

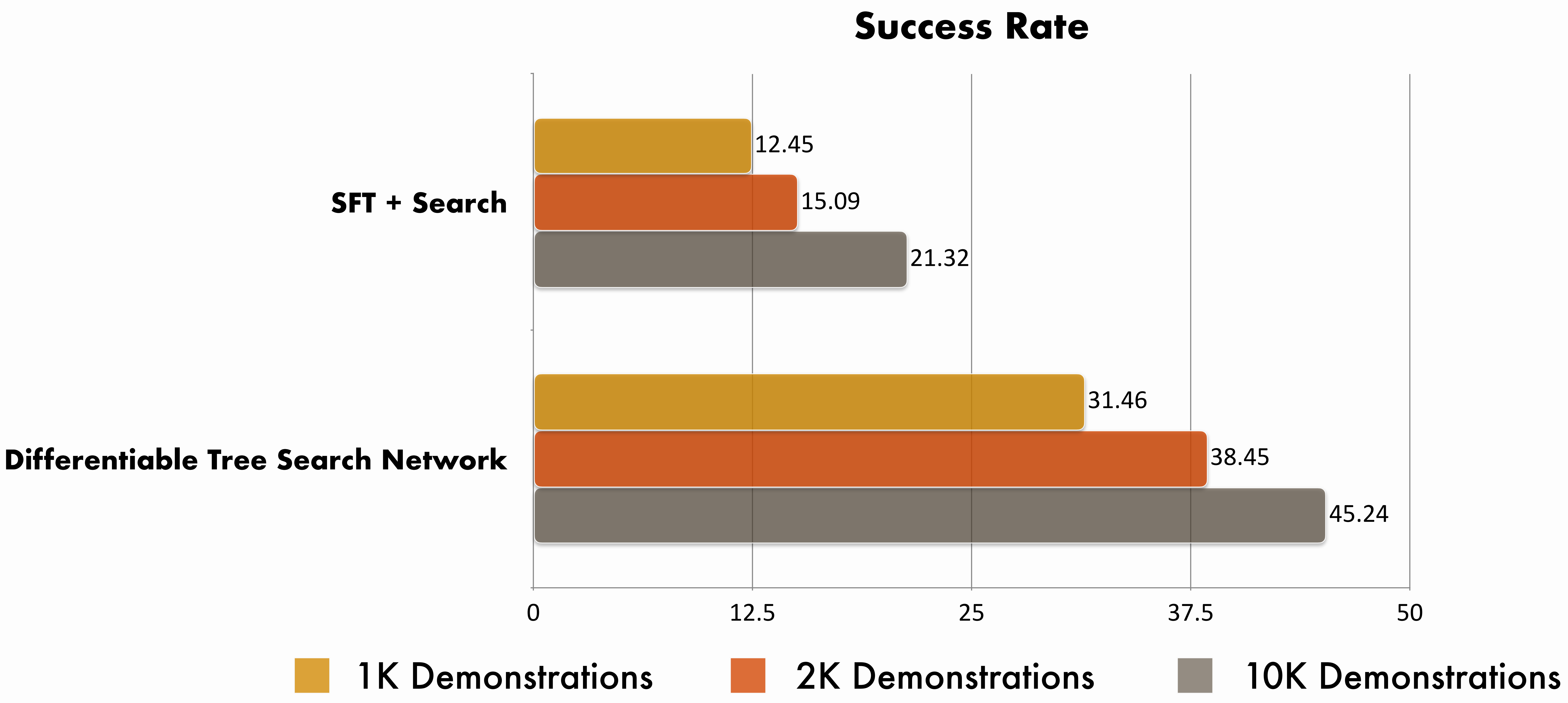
Shallow Tree Search + Joint Optimisation

DOMAIN: **GAME OF 24**

- **Known** world model
- Actions are: Select **operands** or **operators**
- **527** problems for training

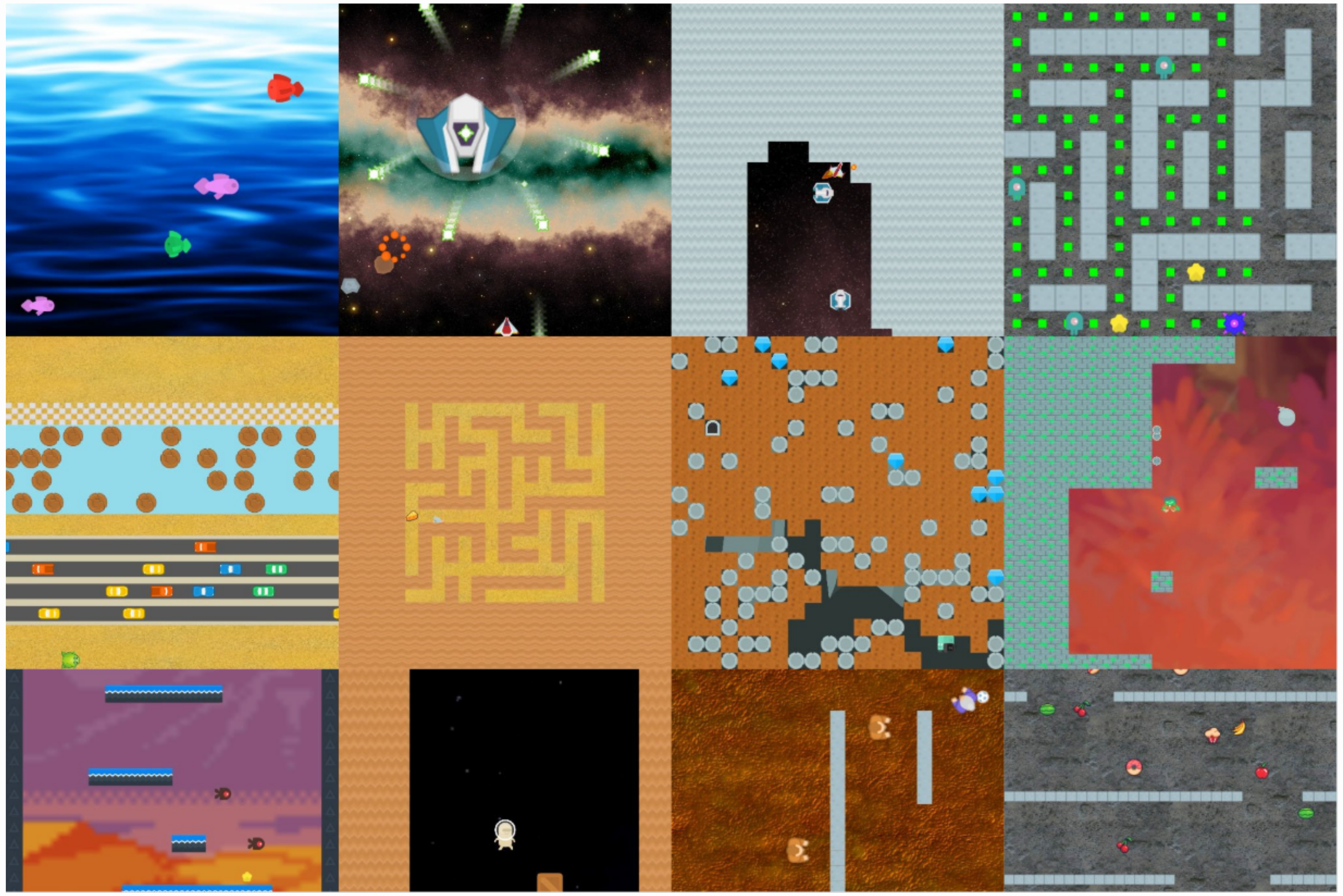


GAME OF 24: **RESULTS**



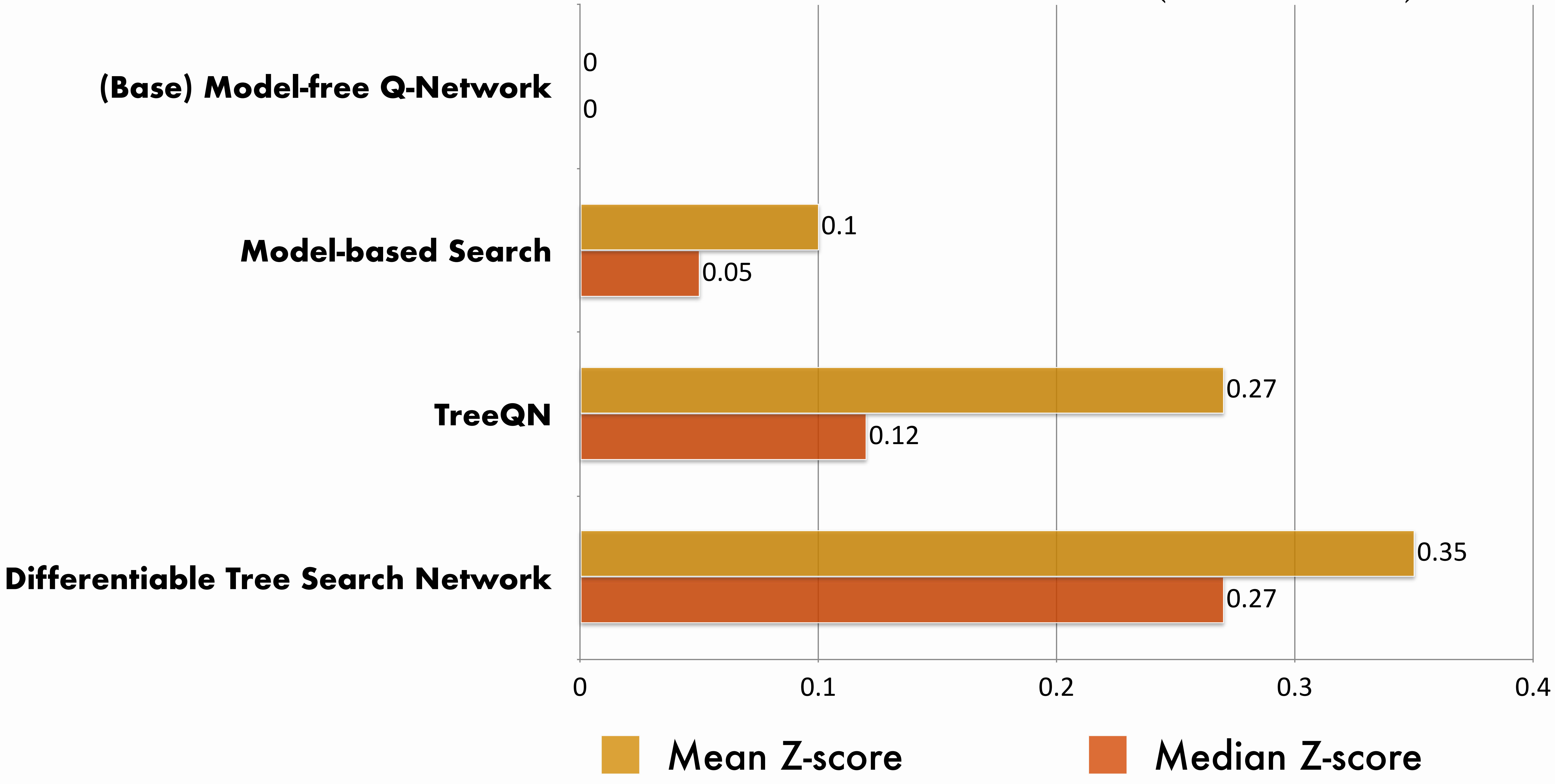
DOMAIN: **PROCGEN**

- Procedurally generated environments
- **16** different games
- Designed to test **generalisation capability** of an agent's policy
- Collected **1000 Trajectories** for training



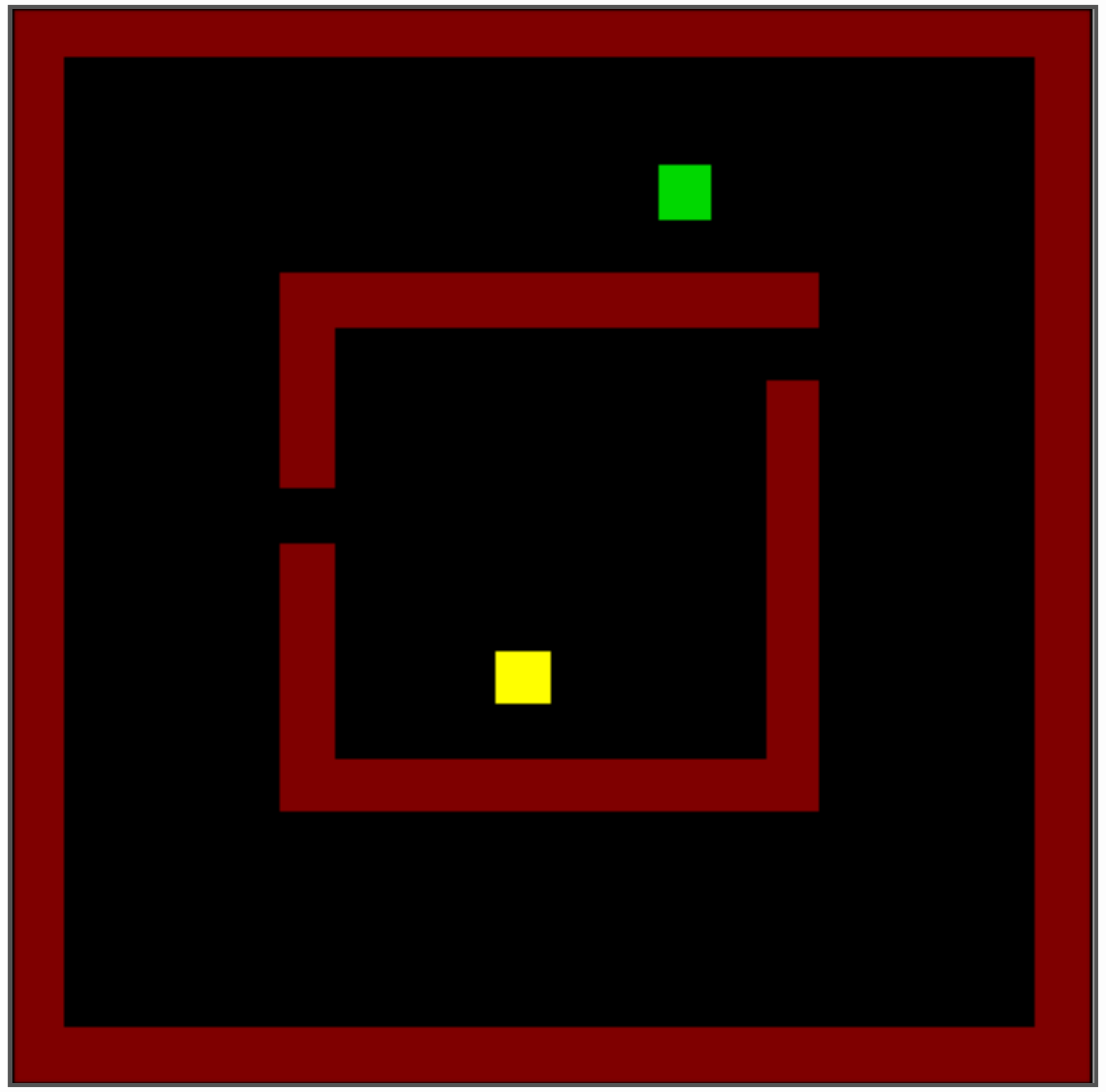
PROCGEN: **RESULTS**

Z-score $\left(Z = \frac{\mu_{agent} - \mu_{base}}{\sigma_{base}} \right)$

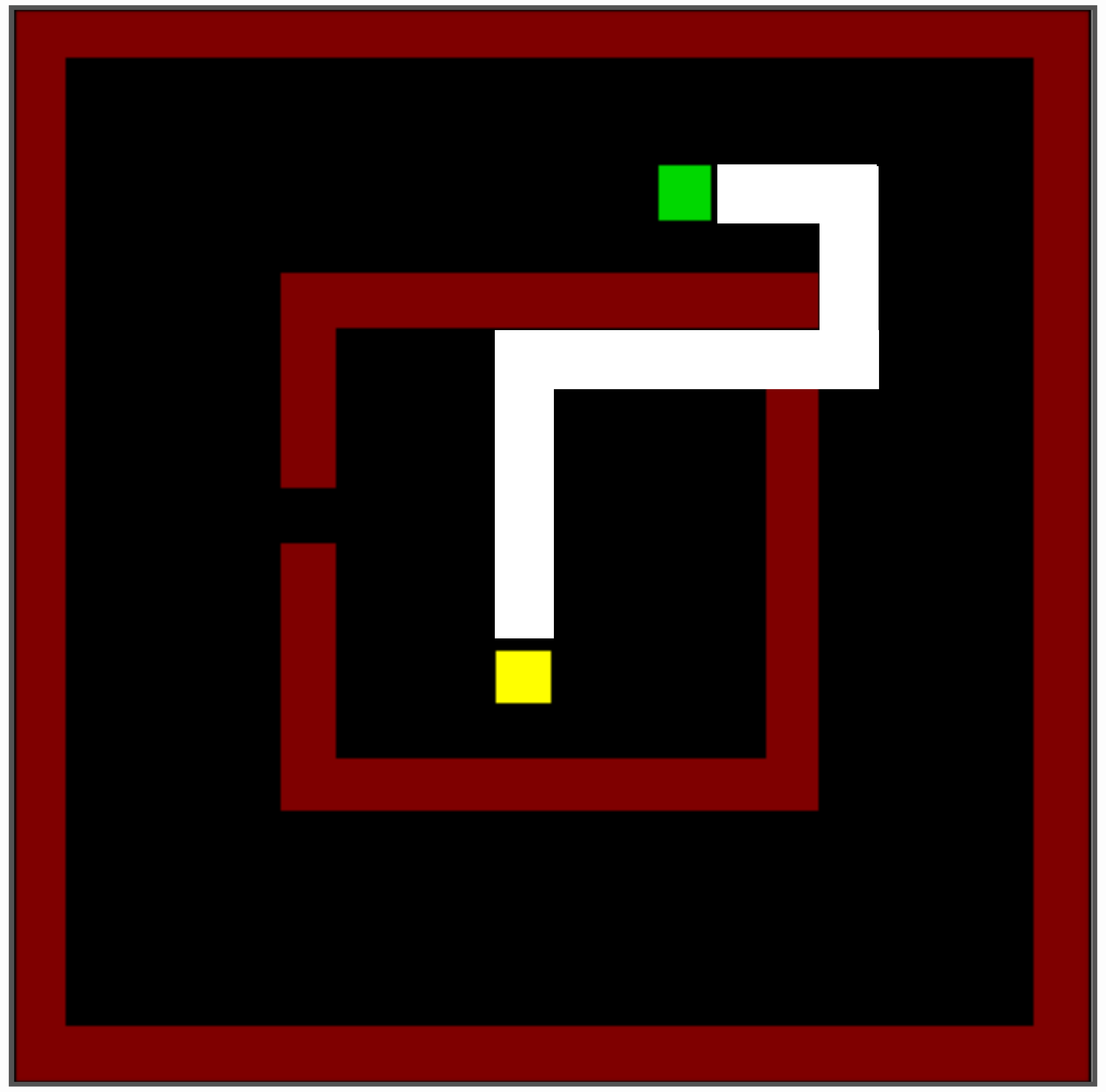


DOMAIN: **NAVIGATION**

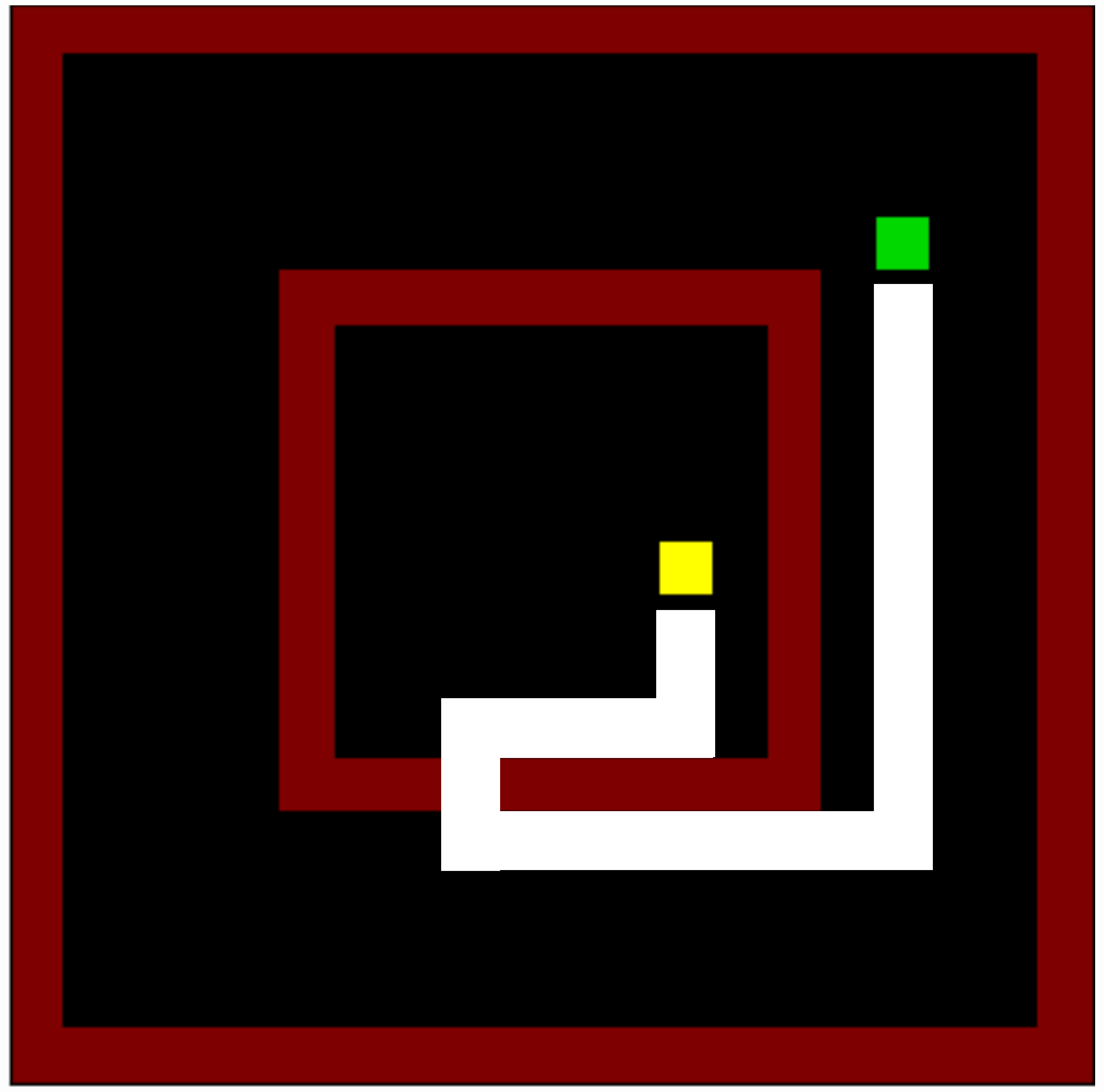
- A simple 10 x 10 grid with 2 areas, i.e. central room and hallway.
- Starts inside the central room and Goal is in the hallway
- Random **Exits x2**
- Collected **1000 Trajectories** for training



NAVIGATION: 2 EXITS VS 1 EXIT (GENERALISATION)

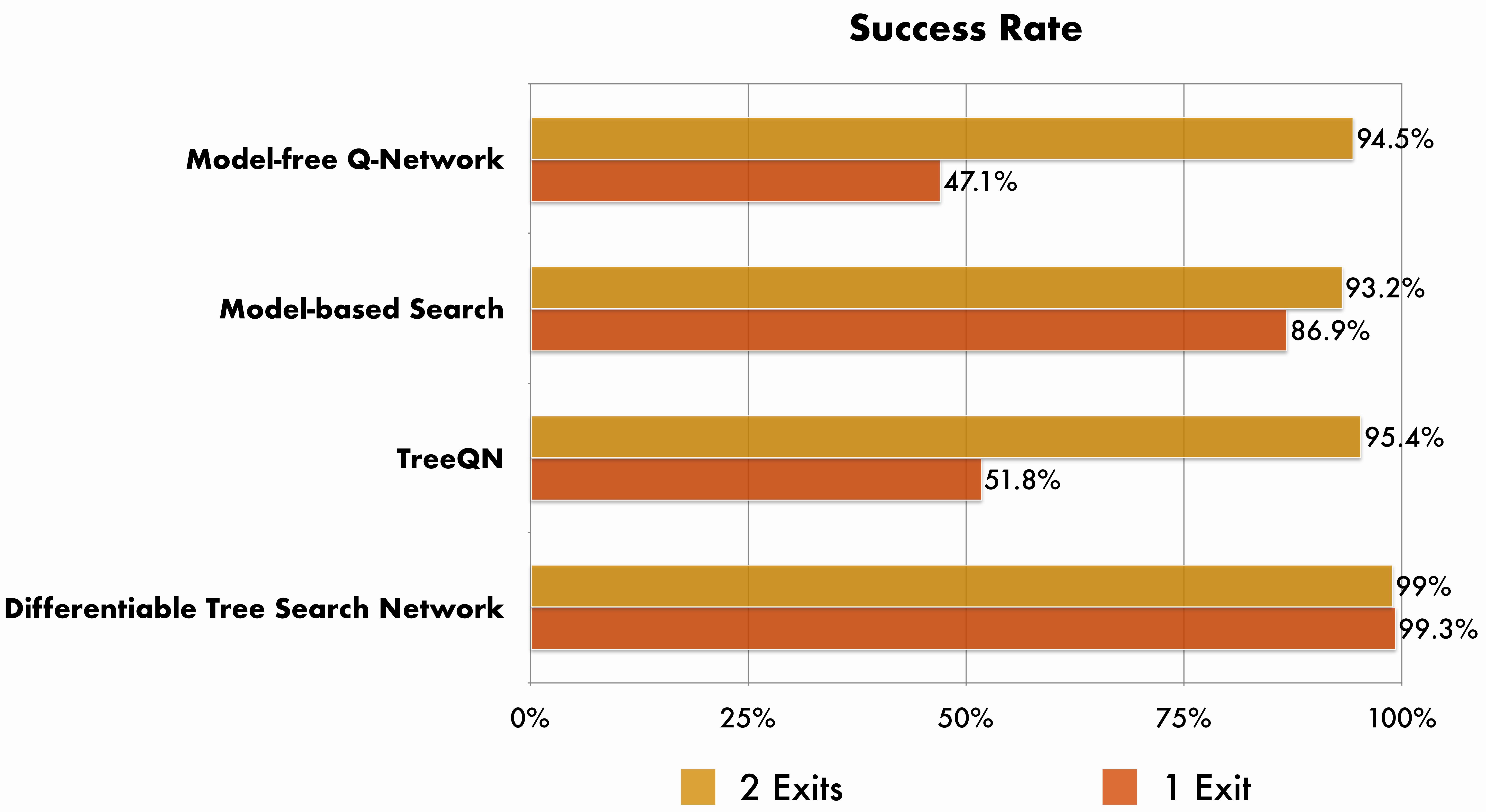


Trained on 2 Exits



Tested on 1 Exit

NAVIGATION: RESULTS



SUMMARY OF CONTRIBUTIONS

Differentiable Tree Search Network (D-TSN)

Modular Neural Network Architecture
that comprises of several *learnable submodules*

Algorithmic Inductive Bias
of a flexible and scalable *best-first tree search* algorithm

Learnt World Model
that is trained to be *useful* for the online search, even if inaccurate

Joint Optimisation
Trains the *search* and *world model* submodules jointly

Additional Technical Details
like *Tree Expansion Policy* and *Telescopic Sum for Variance Reduction*

Thank You!