Reinforcement Learning in Transportation

Peyman Noursalehi

Recap

- Tabular Q-learning algorithm:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \Big]$$

Updated Q-value

Current Q-value

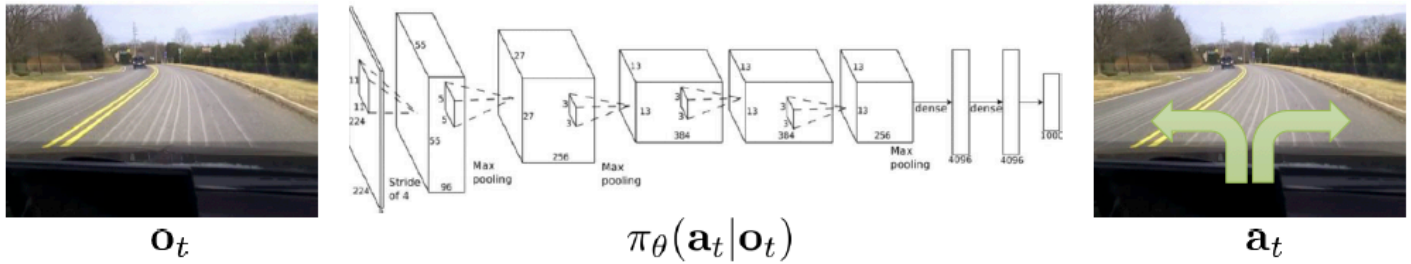
Learning rate

Observed reward

Discount factor

Approximate solutions

- We are often in interested in problems where the state-action combinations are combinatorial
  - State includes
    - Images
    - Continuous variables
    - Large number of features

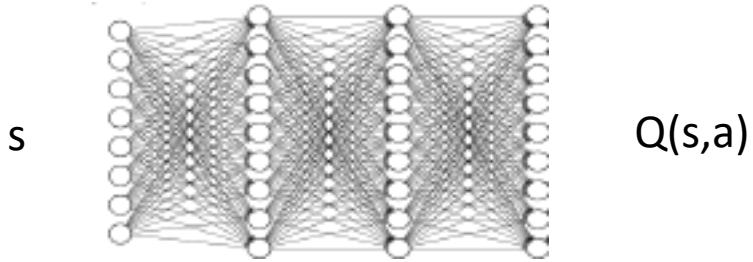


$$200 * 200$$

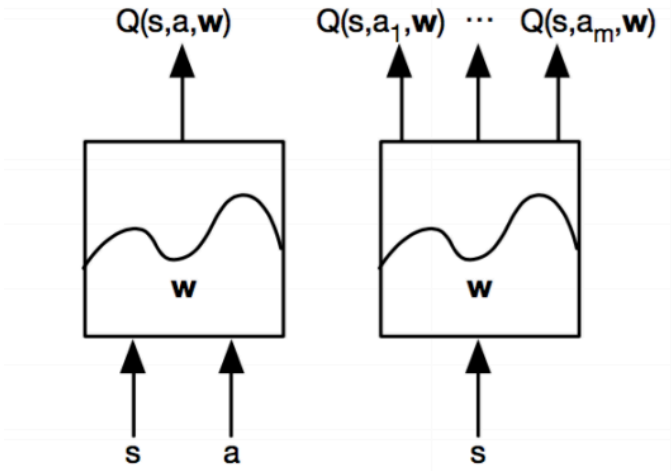
$$|S| = (256)^{200*200}$$

- In such situations, we cannot use a table to hold every possible state-action combination and its value
- Store parameterized state (or state-action) value functions!

Deep Q-Networks



- Use DL to approximate the Q function



- Best suited for discrete action space
- In supervised learning, we had features and labels
  - Straightforward to fit a DL model
- What are our labels here now? How can we convert an RL problem into a supervise learning one?

RL to supervised learning

- Tabular case : $$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\Big]$$

- Since we don't have a table anymore, we can't do direct assignment

- Set $$y_t = R_{t+1} + \gamma \max_a Q_\phi(S_{t+1}, a)$$

- Then *minimize* $$||Q_\phi(S_t, a) - y_t||^2$$

Vanilla DQN algorithm

1. Take an action A_i and observe (S_i, A_i, S_i', R_i)

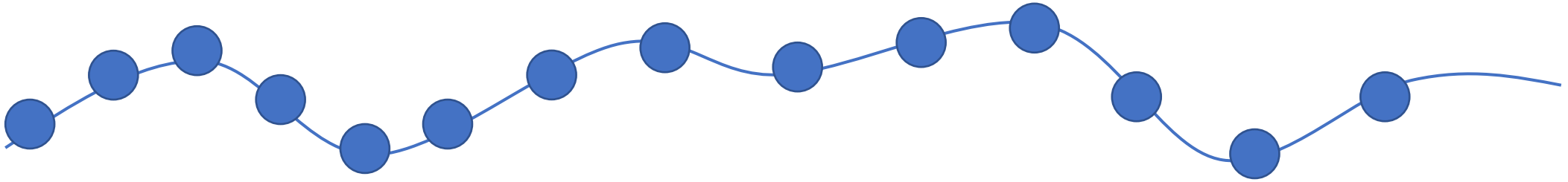
    1. Set $y_t = R(S_t, A_i) + \gamma \max_a Q_\phi(S_{t+1}, a)$
    2. Set $\phi = arg\,min_\phi ||Q_\phi(S_t, a) - y_t||^2$

K

N

Source: Sergey Levine

Issues with the vanilla DQN



Samples are highly correlated

- Violates IID assumption
- Poor generalization

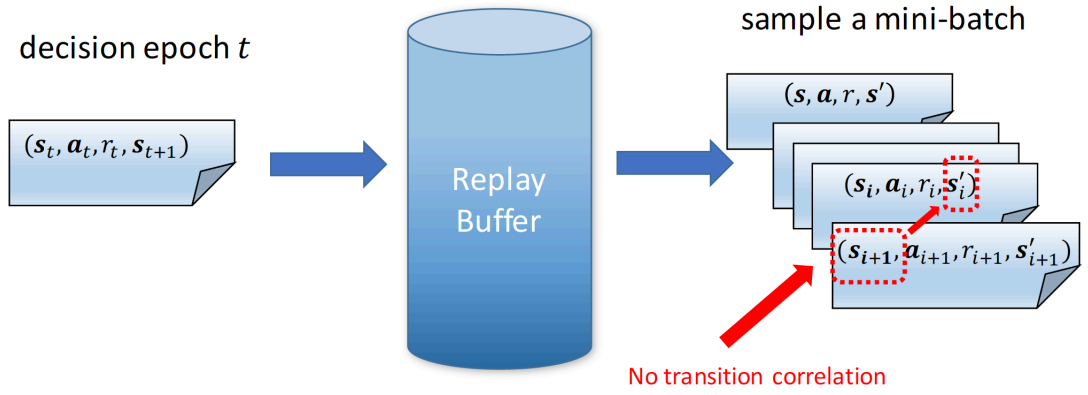
Forgets the disastrous mistakes made before

Source: Sergey Levine

Experience replay

- Store transitions (S, A, R, S') in a buffer
- Sample a mini batch from the replays
- Fit the DL model

s_1, a_1, r_2, s_2
s_2, a_2, r_3, s_3
s_3, a_3, r_4, s_4
...
$s_t, a_t, r_{t+1}, s_{t+1}$

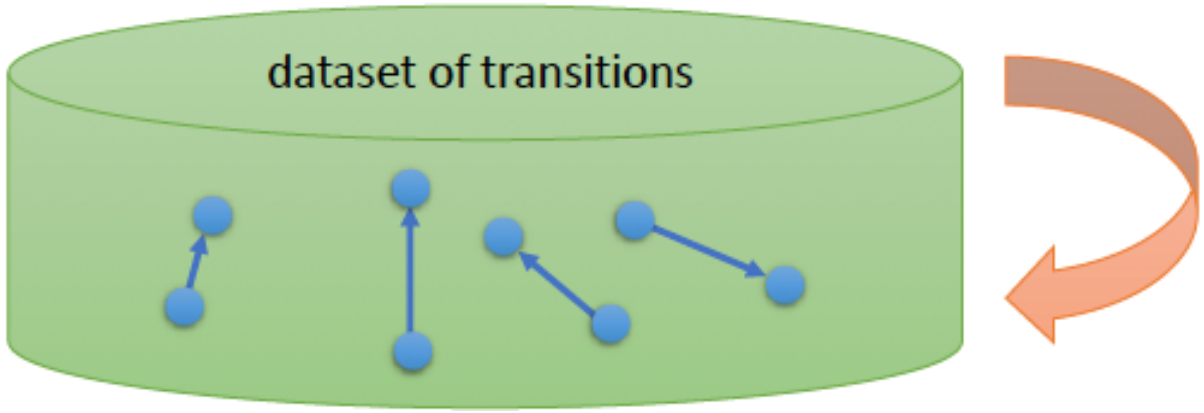


DQN algorithm with experience replay

1. Take an action A_i and observe (S_i, A_i, S_i', R_i) , add it to the buffer

   1. Sample a minibatch from $\{(S_i, A_i, S_i', R_i)\}$
   2. Set $y_i = R(S_i, A_i) + \gamma \max_a Q_\phi(S_i', a)$
   3. Set $\phi = arg\,min \sum_i ||Q_\phi(s_i, a_i) - y_i||^2$

N K



Source: Sergey Levine

Issues with the vanilla DQN

- Samples are highly correlated → Use a replay buffer
- Chasing a moving target

$$y_t = R(S_t, A_i) + \gamma \max_a Q_\phi(S_{t+1}, a)$$

$$\phi = arg\,min_\phi ||Q_\phi(S_t, a) - y_t||^2$$

Divergence in DQN without Target Networks

① At first everything will look normal. We just chase the target.

② But the target will move as our Q-function improves.

③ Then, things go bad.

④ An the non-stationary can make us diverge.

DQN with target networks

① By freezing the target 	② We make progress towards it before 
③ we update it, and change it. This makes the algorithm more stable 	④ And able to get good results 

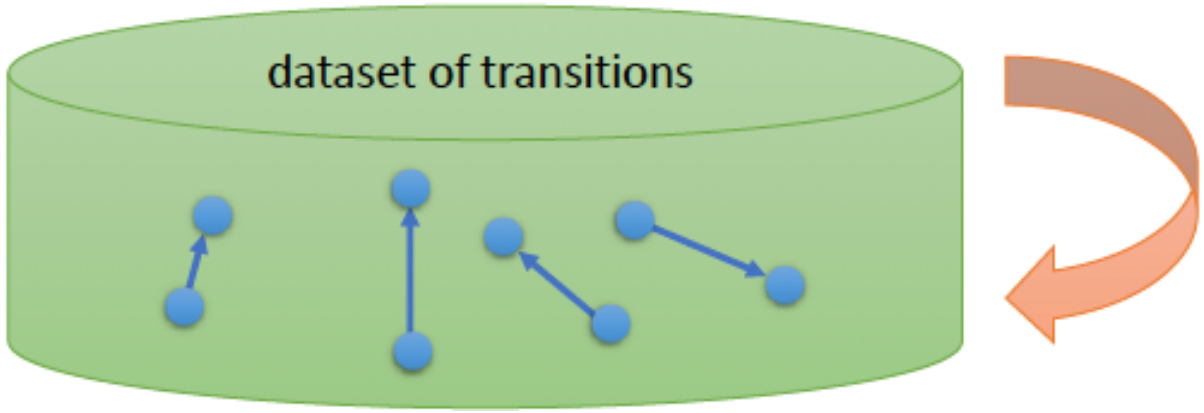
DQN algorithm with target network

1. Take an action A_i and observe (S_i, A_i, S_i', R_i) , add it to the buffer

   1. Sample a minibatch from $\{(S_i, A_i, S_i', R_i)\}$
   2. Set $y_i = R(S_i, A_i) + \gamma \max_a Q_\phi'(S_i', a)$
   3. Set $\phi = arg\,min \sum_i ||Q_\phi(s_i, a_i) - y_i||^2$

K

N



Source: Sergey Levine

Target network

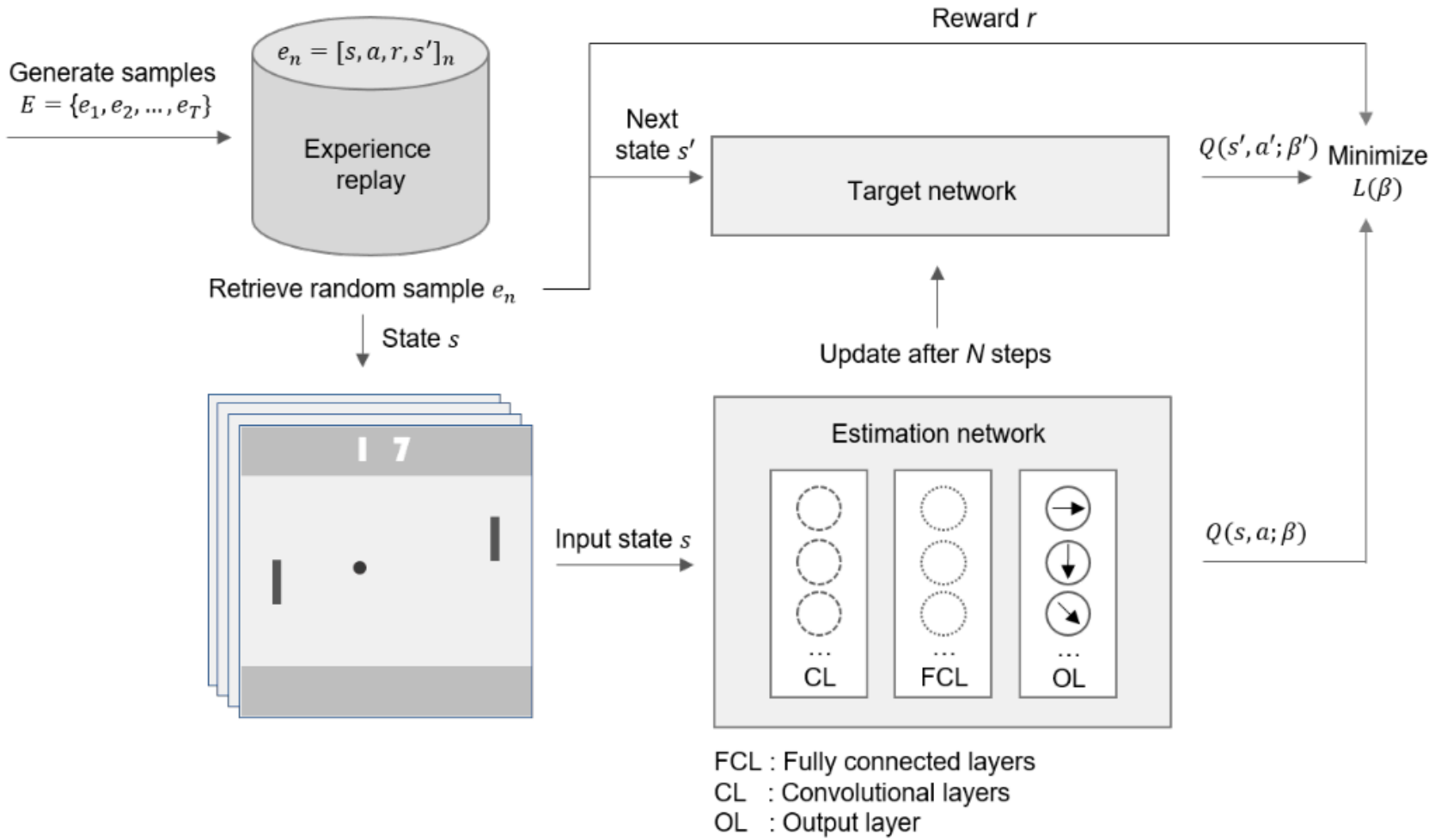
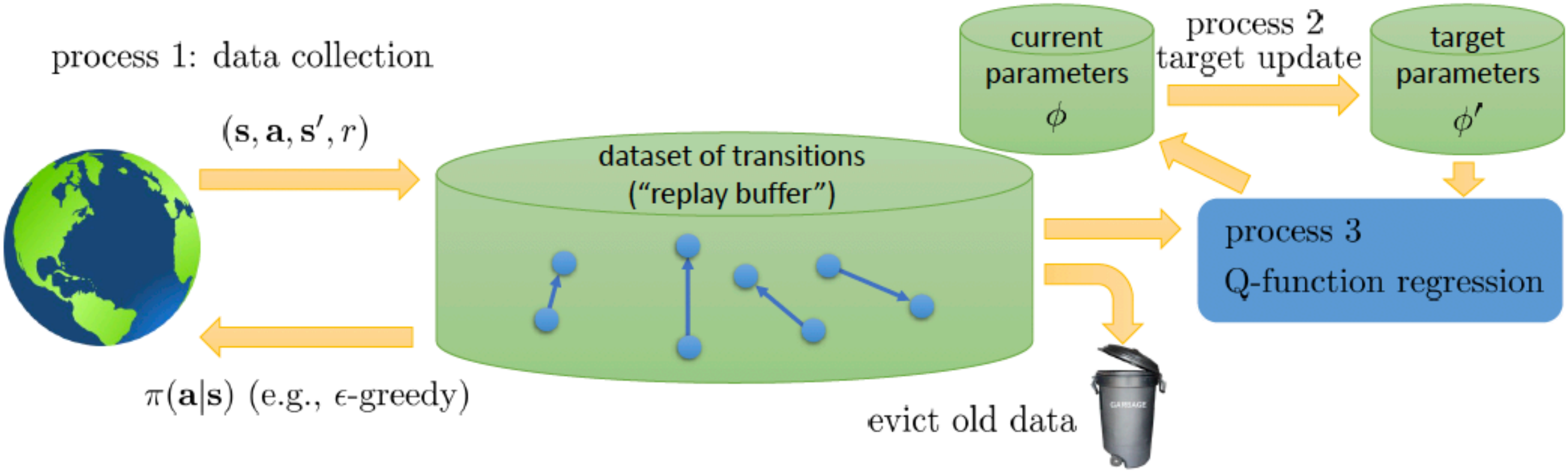


Fig. 5: Deep Q-network structure.

A more general view



Credit: Sergey Levine

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

DQN for playing Atari games

- Stacked CNN layers for the Q-function

- Trick for handling the temporal dependencies:
  - Stack 4 images together

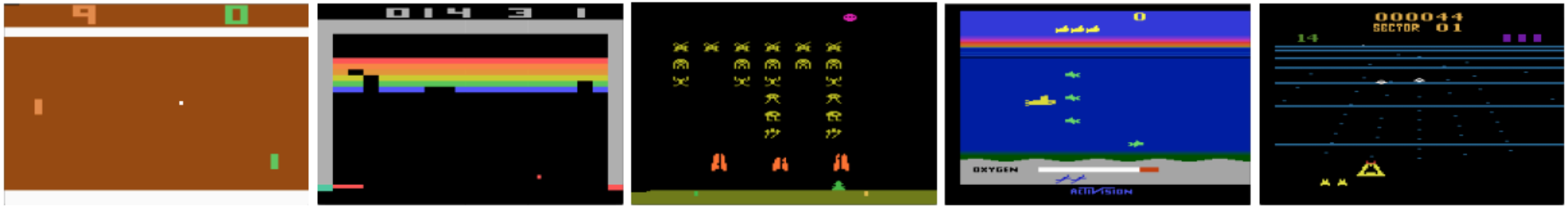
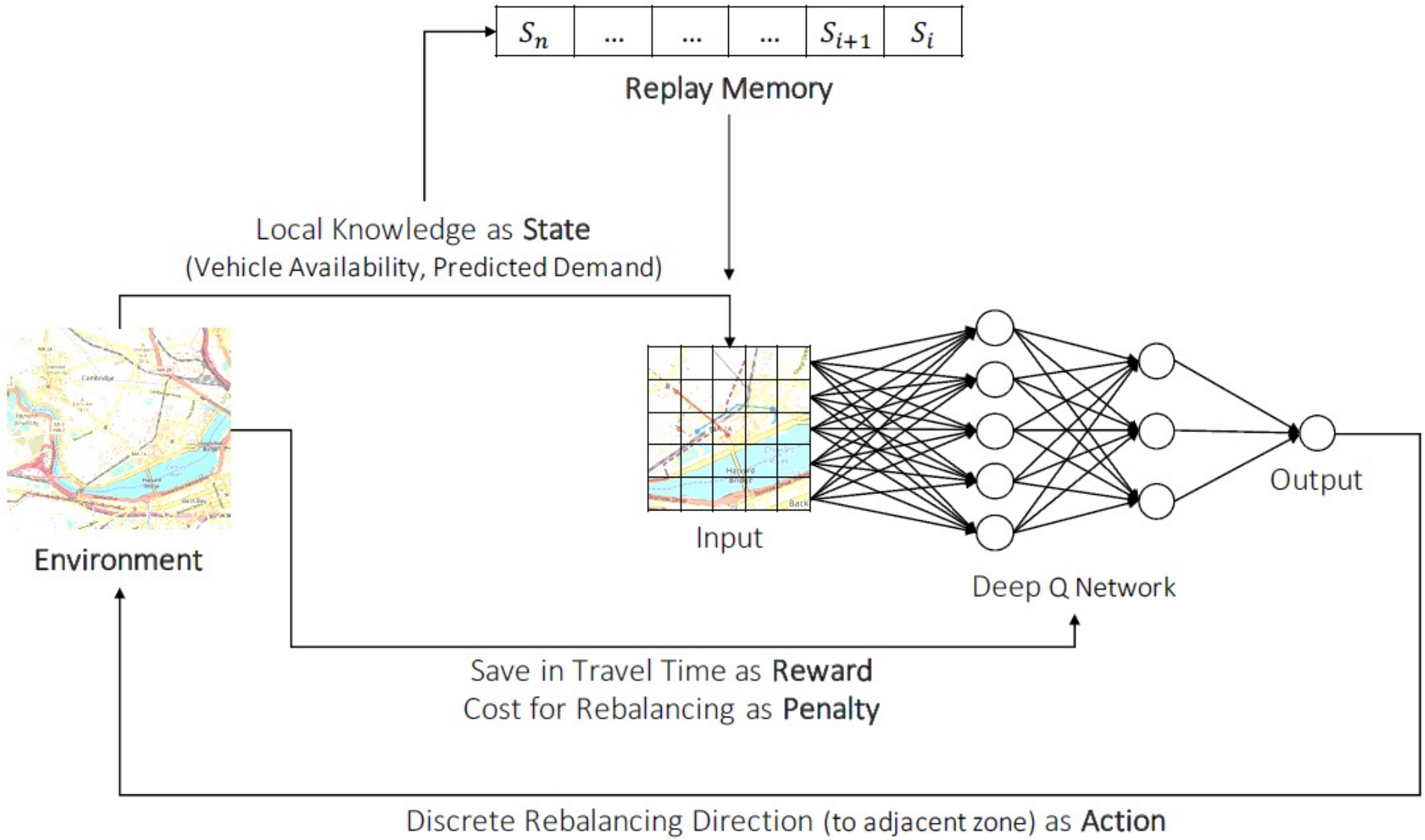
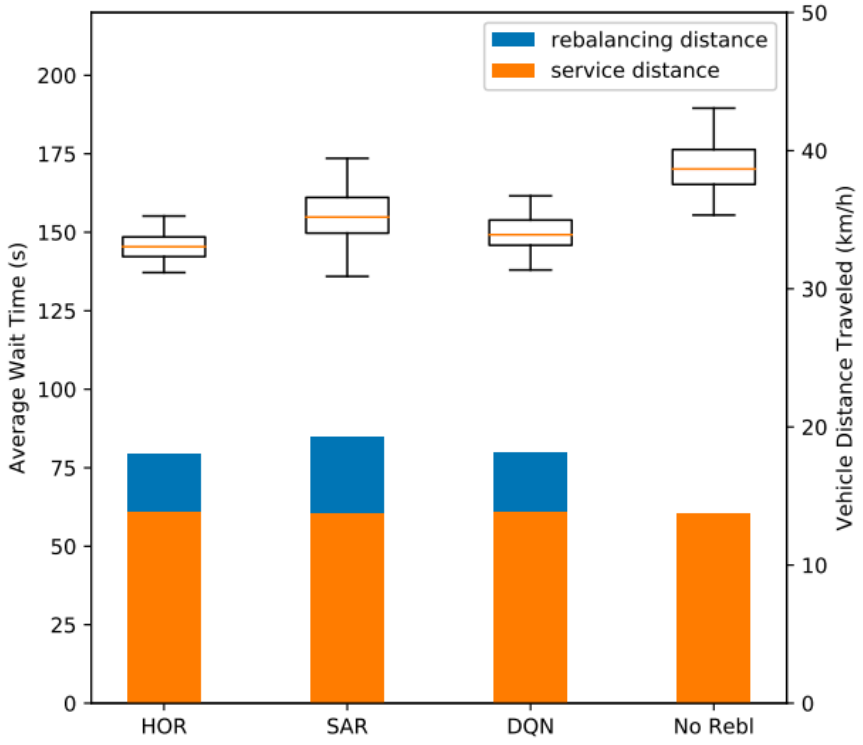


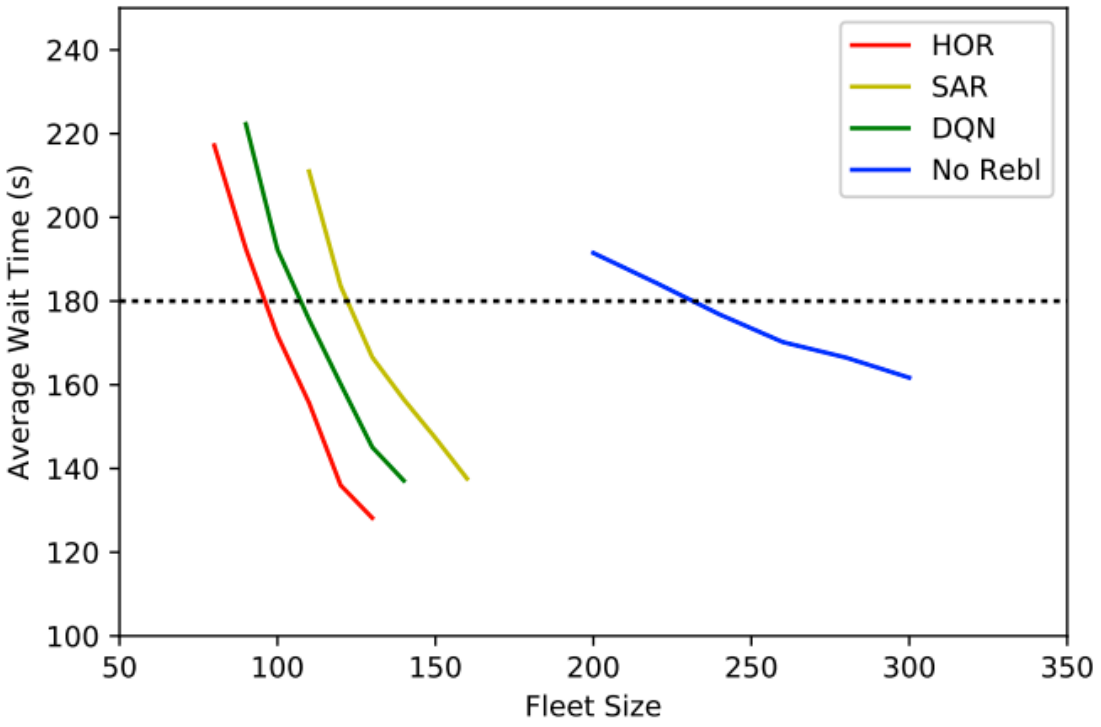
Figure 1: Screen shots from five Atari 2600 Games: (*Left-to-right*) Pong, Breakout, Space Invaders, Seaquest, Beam Rider

Rebalancing Shared Mobility-on-Demand Systems: a Reinforcement Learning Approach

Jian Wen*, Jinhua Zhao† and Patrick Jaillet‡
* Department of Civil and Environmental Engineering
† Department of Urban Studies and Planning
‡ Department of Electrical Engineering and Computer Science
Laboratory for Information and Decision Systems, Operations Research Center
Massachusetts Institute of Technology, Cambridge, MA 02139-4307
Emails: wenj@mit.edu, jinhua@mit.edu, jaillet@mit.edu


S_n ... ... S_{i+1} S_i
Replay Memory
Local Knowledge as State
(Vehicle Availability, Predicted Demand)
Environment
Input
Output
Deep Q Network
Save in Travel Time as Reward
Cost for Rebalancing as Penalty
Discrete Rebalancing Direction (to adjacent zone) as Action





Rebalancing induced around 30% more empty miles

Spatio-Temporal Capsule-based Reinforcement Learning for Mobility-on-Demand Network Coordination

Suining He
The University of Michigan–Ann Arbor
suiningh@umich.edu

Kang G. Shin
The University of Michigan–Ann Arbor
kgshin@umich.edu

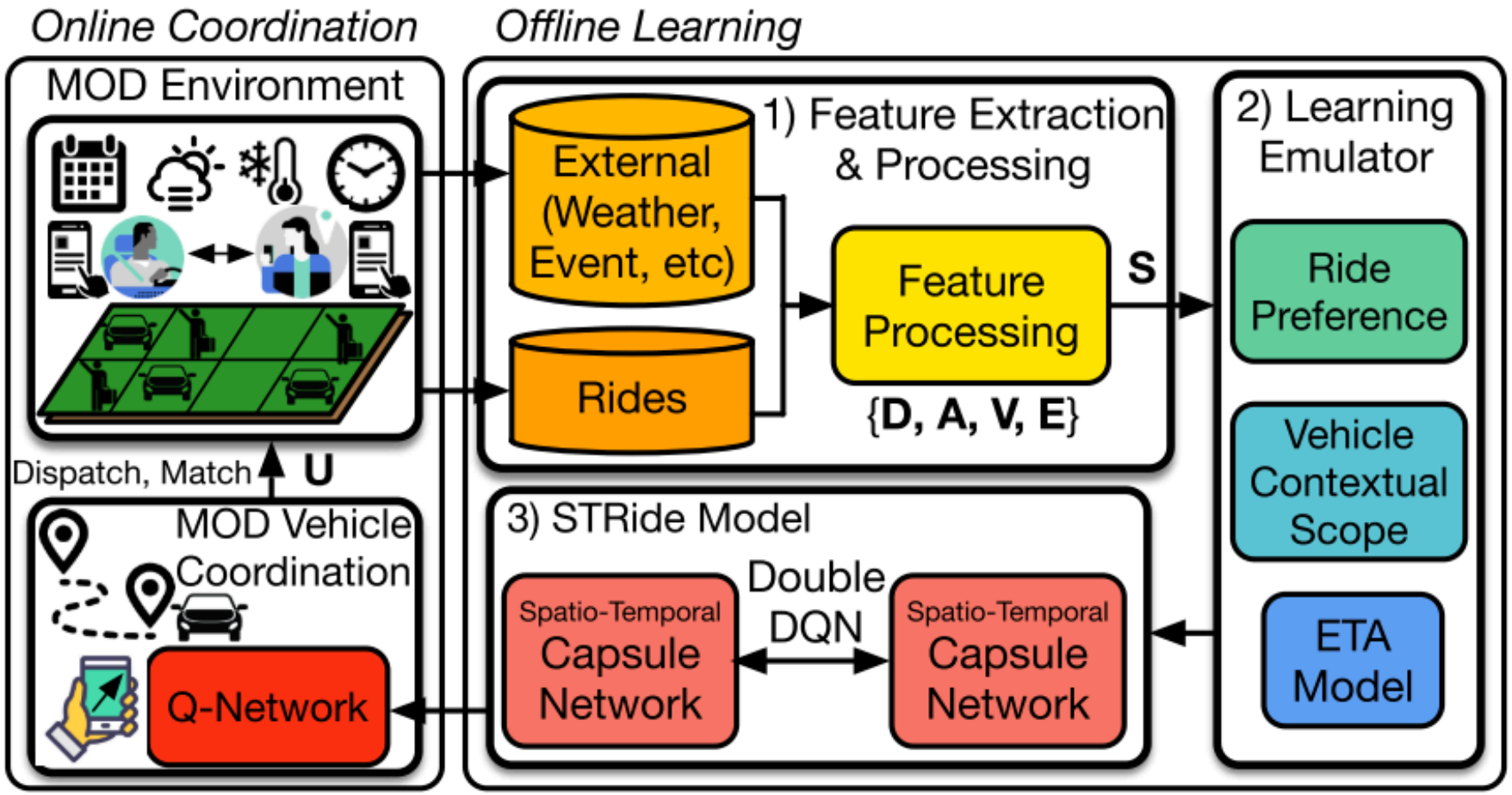


Fig. 1: The system framework of STRide.

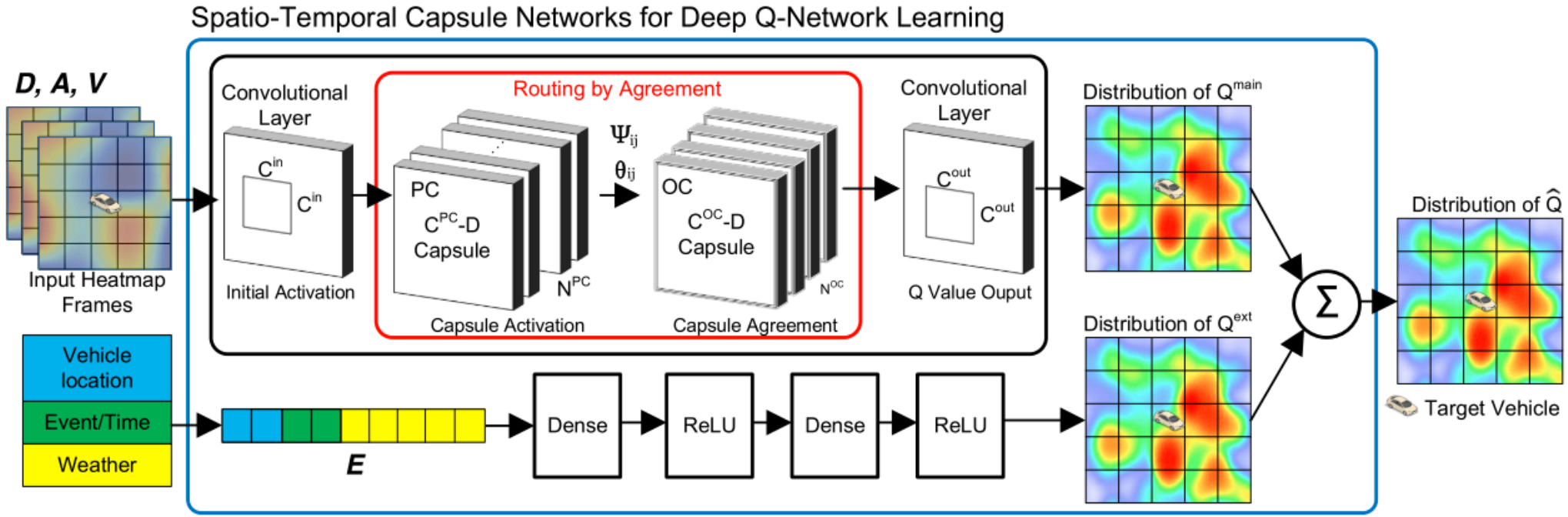
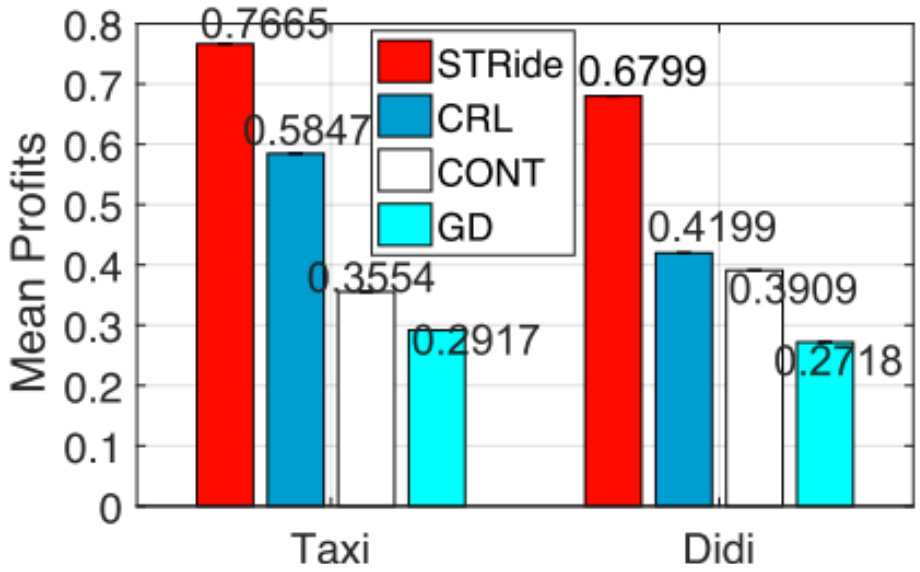


Fig. 2: Core module of spatio-temporal capsule-based Q-netwo



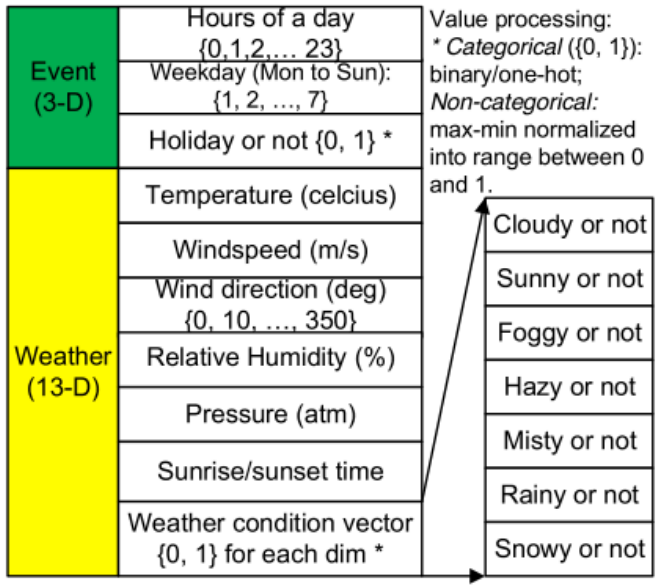


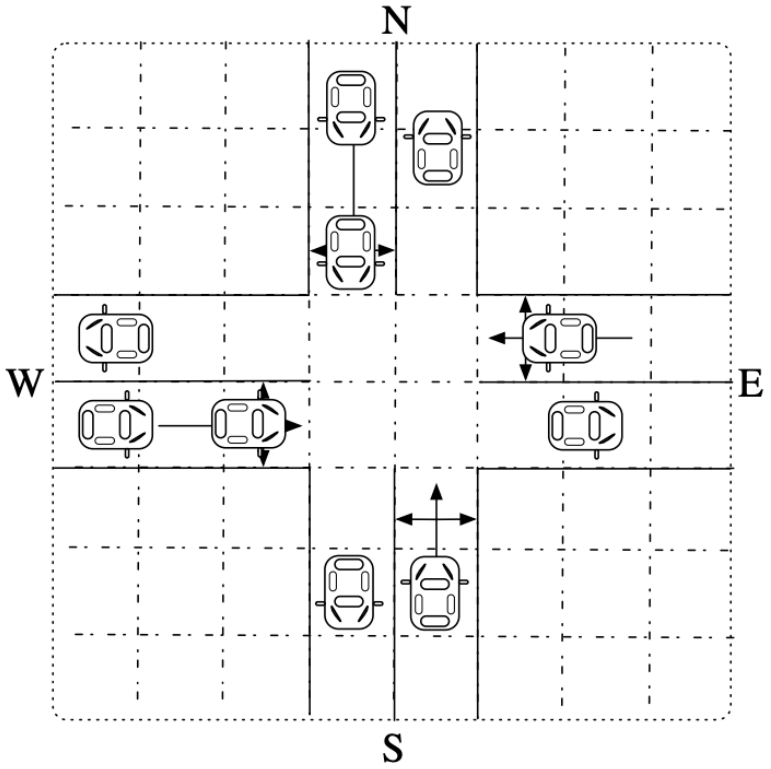
Fig. 3: External factors E.

Deep Reinforcement Learning for Traffic Light Control in Vehicular Networks

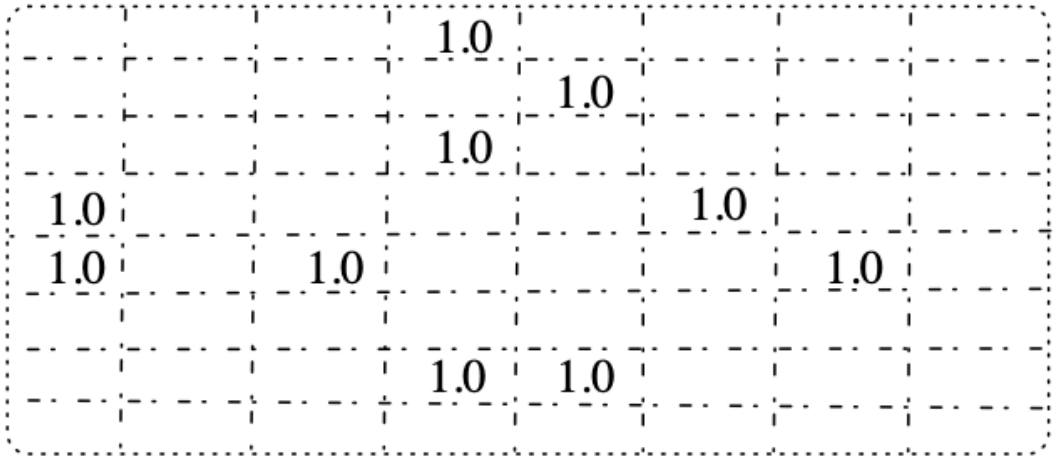
Xiaoyuan Liang, Xusheng Du, *Student Member, IEEE,* Guiling Wang, *Member, IEEE,* and Zhu Han *Fellow, IEEE*

Problem formulation

- State: [position, speed]



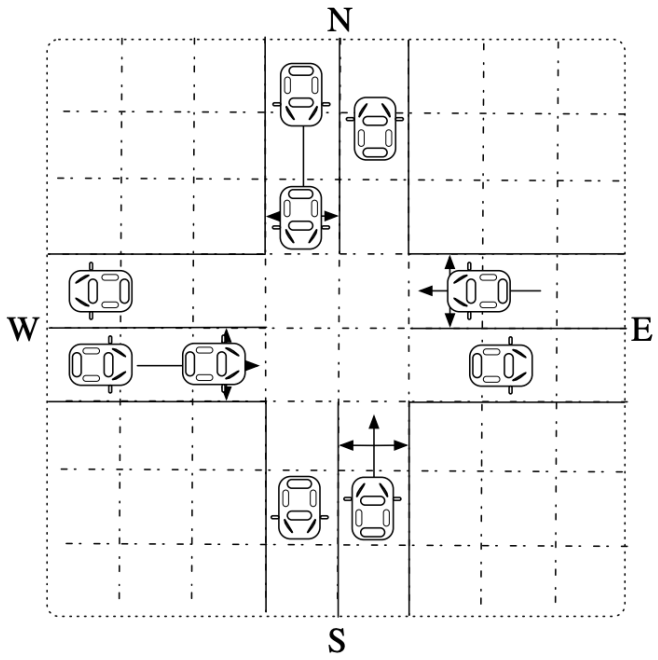
(a) The snapshot of traffic on a road at one moment



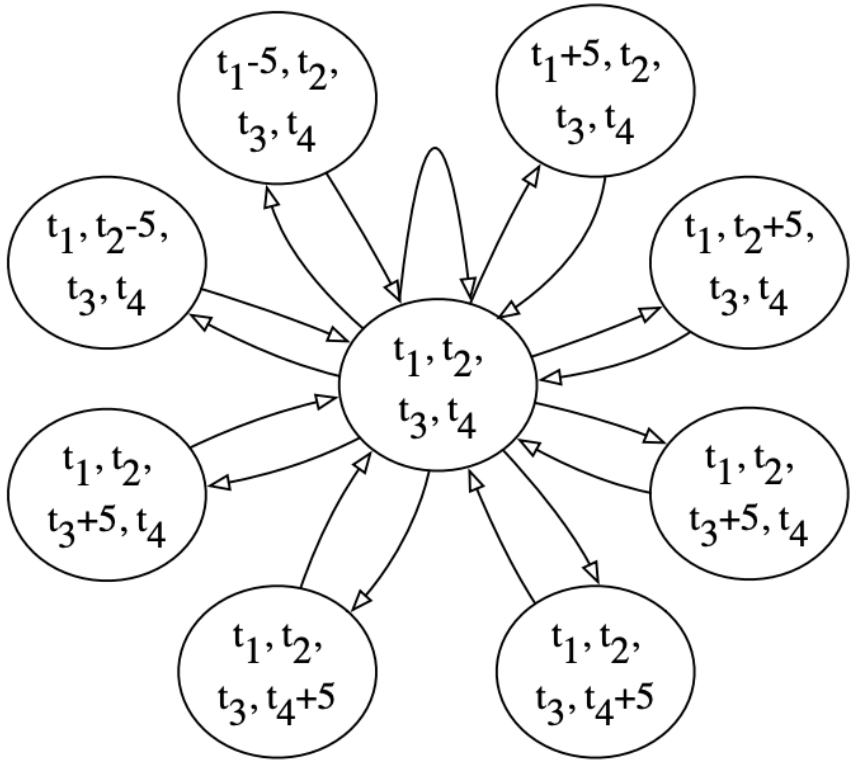
(b) The corresponding position matrix on this road

Problem formulation

- Action Space
- $< t_1, t_2, t_3, t_4 >$ duration of 4 phases

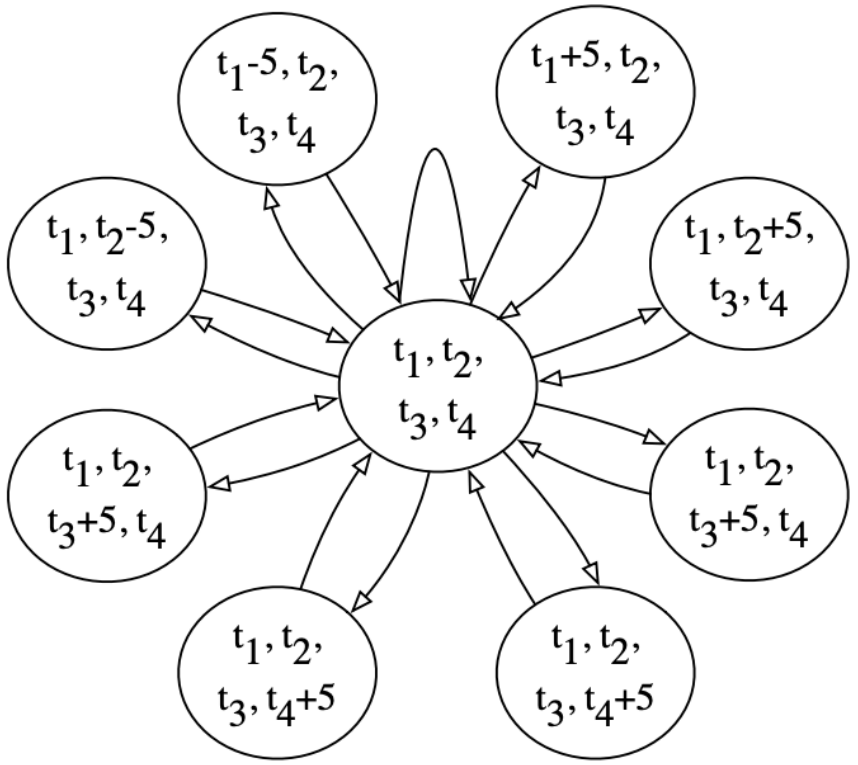


(a) The snapshot of traffic on a road at one moment

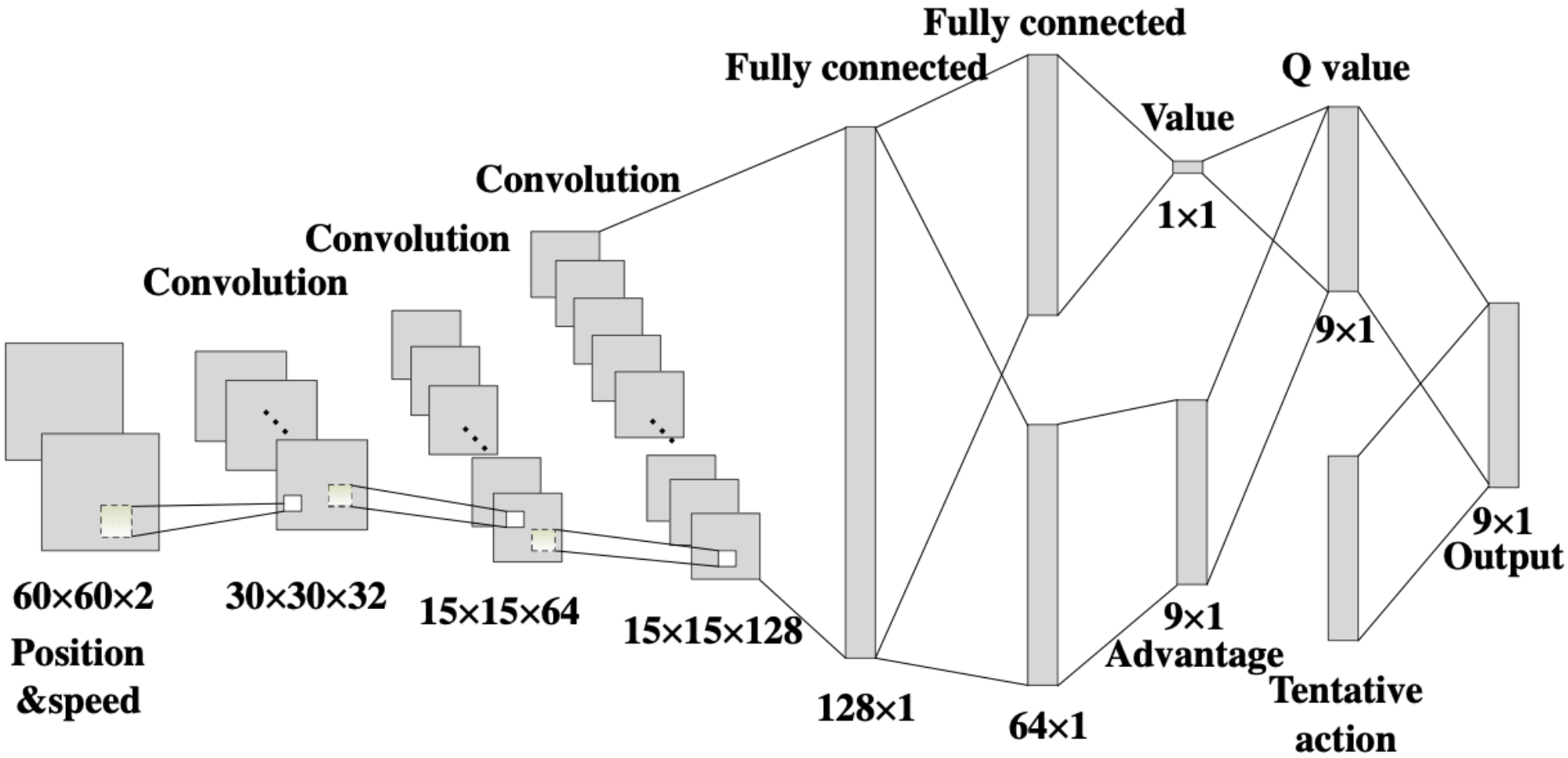


Problem formulation

- Action Space
- $< t_1, t_2, t_3, t_4 >$ duration of 4 phases
- Reward:
  - The change in cumulative waiting time between two neighboring cycles



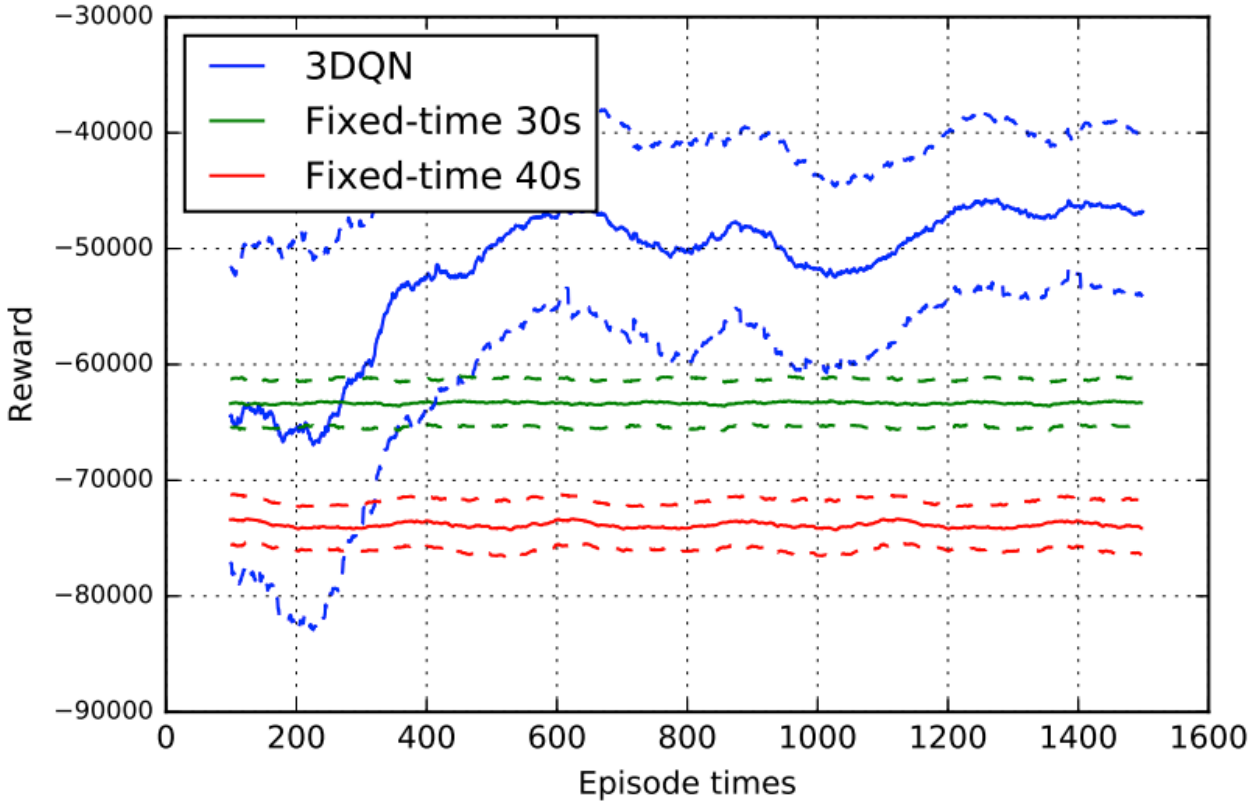
Network architecture



Results

PARAMETERS IN THE REINFORCEMENT LEARNING NETWORK

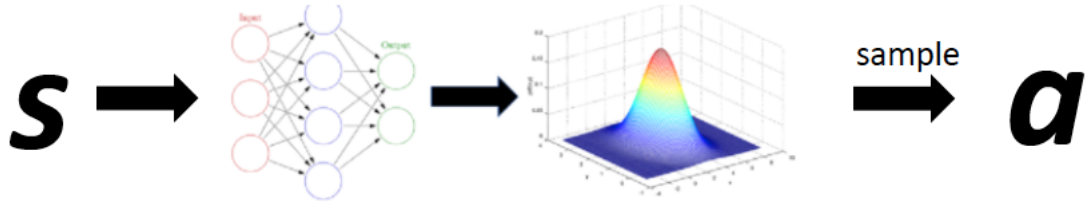
Parameter	Value
Replay memory size M	20000
Minibatch size B	64
Starting ϵ	1
Ending ϵ	0.01
Steps from starting ϵ to ending ϵ	10000
Pre-training steps tp	2000
Target network update rate α	0.001
Discount factor γ	0.99
Learning rate ϵ_r	0.0001
Leaky ReLU β	0.01



Policy gradient methods

- Directly learn the policy function
- Why?
  - Effective in high-dimensional action space
  - Easy to use in continuous action space
  - Can learn stochastic policies
- The policy directly gives us a probability over actions for each state

$$y_i = R(S_i, A_i) + \gamma \max_a Q_\phi(S_i', a)$$



Credit: Cathy Wu

Objective function in RL

Maximize the overall reward in a sequential problem

$$J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[\sum_t r(s_t, a_t) \right]$$

Where

$$\pi_\theta(\tau) = p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)$$

Policy evaluation:

$$J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[\sum_t r(s_t, a_t) \right] \approx \frac{1}{N} \sum_t \sum_i r(s_{i,t}, a_{i,t})$$

Direct policy differentiation

$$\theta^* = arg\max_\theta J(\theta) = arg\max_\theta E_{\tau\sim\pi_\theta(\tau)}\left[\sum_t r(s_t, a_t)\right]$$

$$J(\theta) = E_{\tau\sim\pi_\theta(\tau)}\left[r(\tau)\right] = \int \pi_\theta(\tau) r(\tau) d\tau$$

$$\nabla_\theta J(\theta) = \int \nabla_\theta \pi_\theta(\tau) r(\tau) d\tau$$

Recall:

$$\pi_\theta(\tau) = p(s_1)\prod_{t=1}^{T}\pi_\theta(a_t|s_t)p(s_{t+1}|s_t, a_t)$$

We don't know this!

Credit: Sergey Levine

Direct policy differentiation

$$\theta^* = arg\max_\theta J(\theta) = arg\max_\theta E_{\tau\sim\pi_\theta(\tau)}\left[\sum_t r(s_t, a_t)\right]$$

$$J(\theta) = E_{\tau\sim\pi_\theta(\tau)}\left[r(\tau)\right] = \int \pi_\theta(\tau) r(\tau) d\tau$$

$$\nabla_\theta J(\theta) = \int \nabla_\theta \pi_\theta(\tau) r(\tau) d\tau$$

$$\pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau) = \pi_\theta(\tau)\frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)} = \nabla_\theta \pi_\theta(\tau)$$

$$\nabla_\theta J(\theta) = \int \pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau) r(\tau) d\tau$$

$$\nabla_\theta J(\theta) = E_{\tau\sim\pi_\theta(\tau)}\left[\nabla_\theta \log \pi_\theta(\tau) r(\tau)\right]$$

Credit: Sergey Levine

Direct policy differentiation

Recall:

$$\pi_\theta(\tau) = p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)$$

$$\nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[\nabla_\theta \log \pi_\theta(\tau) r(\tau) \right]$$

$$\log \pi_\theta(\tau) = \log p(s_1) + \sum_t \log \pi_\theta(a_t|s_t) + \log p(s_{t+1}|s_t, a_t)$$

$$\nabla_\theta \log \pi_\theta(\tau) = \sum_t \nabla_\theta \log \pi_\theta(a_t|s_t)$$

$$\nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[\sum_t \nabla_\theta \log \pi_\theta(a_t|s_t) \right] \left[\sum_t r(s_t, a_t) \right]$$

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i^N \left[\sum_t \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right] \left[\sum_t r(s_{i,t}, a_{i,t}) \right]$$

Credit: Sergey Levine

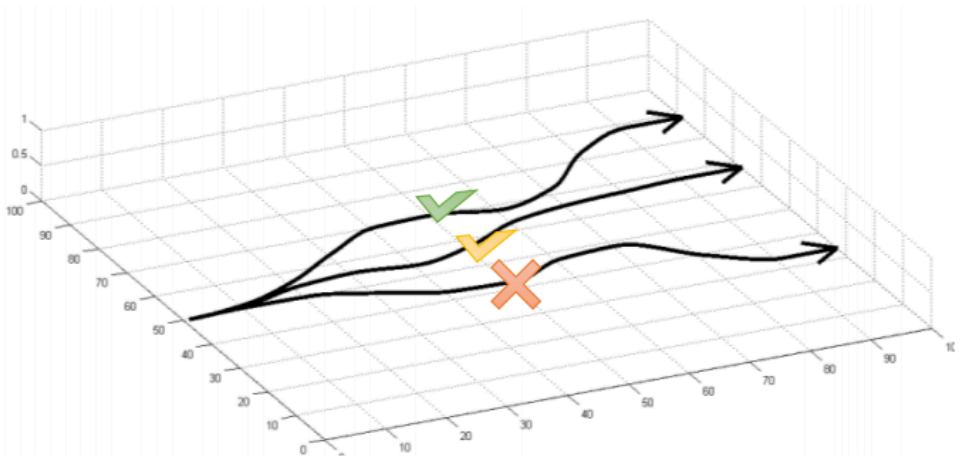
REINFORCE

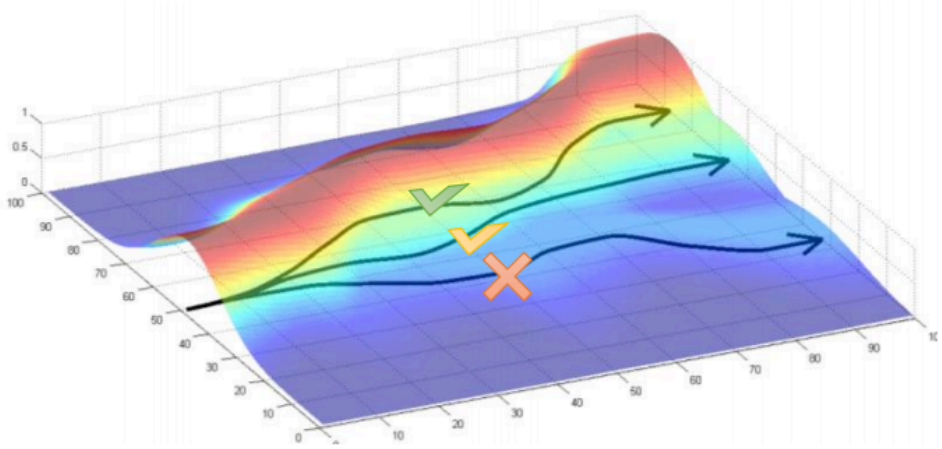
1. Sample $\{\tau^i\}$ from π_θ
2. $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i^N [\sum_t \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})] [\sum_t r(s_{i,t}, a_{i,t})]$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Credit: Sergey Levine

What happened?

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i^N [\sum_t \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})] [\sum_t r(s_{i,t}, a_{i,t})]$$





Make good stuff more likely!

Credit: Sergey Levine

Policy gradients have a high variance

- Gradients are multiplied by the rewards

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i^N [\sum_t \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})] [\sum_t r(s_{i,t}, a_{i,t})]$$

- One ”lucky” episode gets reward of 1000, compared to the avg 700
  - Dominates the gradient
  - Makes it difficult for SGD to learn the optimal weights
  - Training becomes unstable
- What can we do?

Baseline

- Subtract a baseline from the rewards

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i^N \left[\sum_t \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right] \left[\sum_t r(s_{i,t}, a_{i,t}) - b \right]$$

- The simplest baseline is the average reward

$$b = \frac{1}{N} \sum_i^N r(\tau)$$

- Can we do better?

$$V(s_t) = E_{a_t \sim \pi_\theta(a_t|s_t)}[Q(s_t, a_t)]$$

Advantage

- How much better a_t is

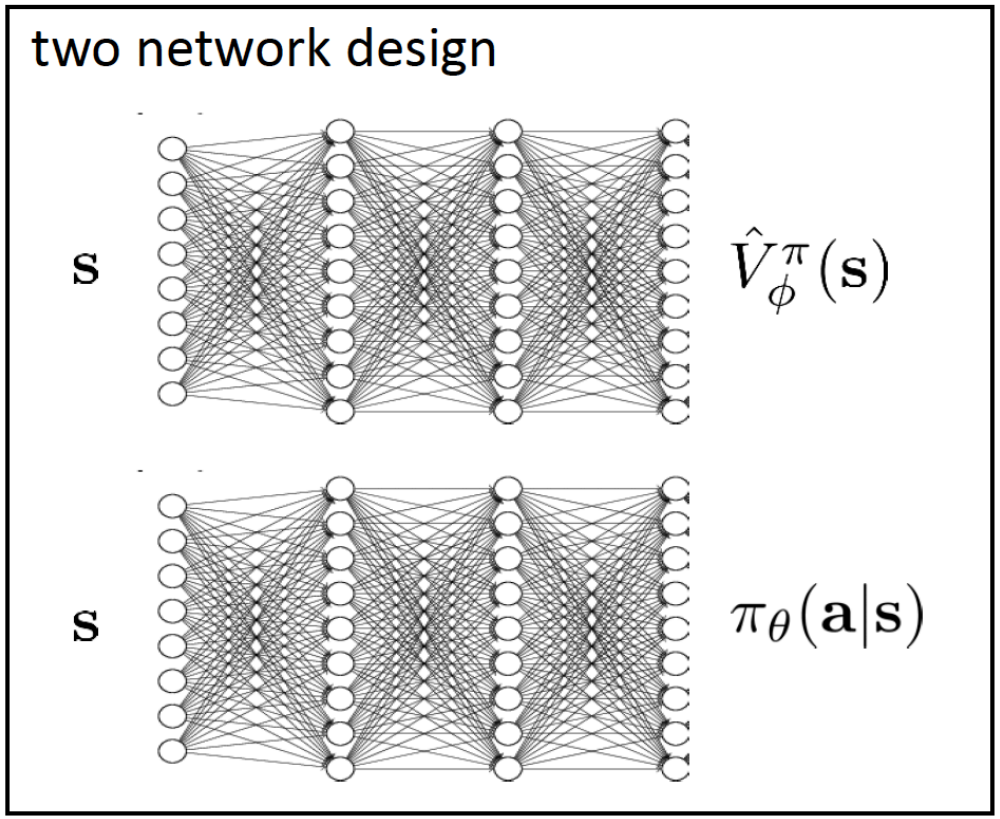
$$A^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t)$$

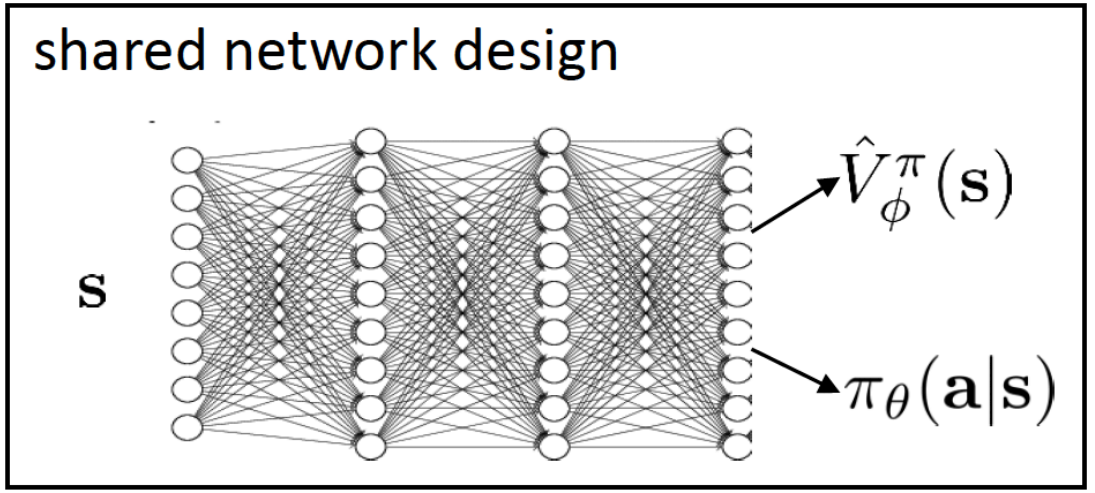
$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i}^{N} \sum_{t} \nabla_{\theta} \log \pi_{\theta}(a_{i,t}|s_{i,t}) A^{\pi}(s_{i,t}, a_{i,t})$$

- Train another DL network to predict $V^{\pi}(s_t)$

Actor-critic models

- Two architectures:





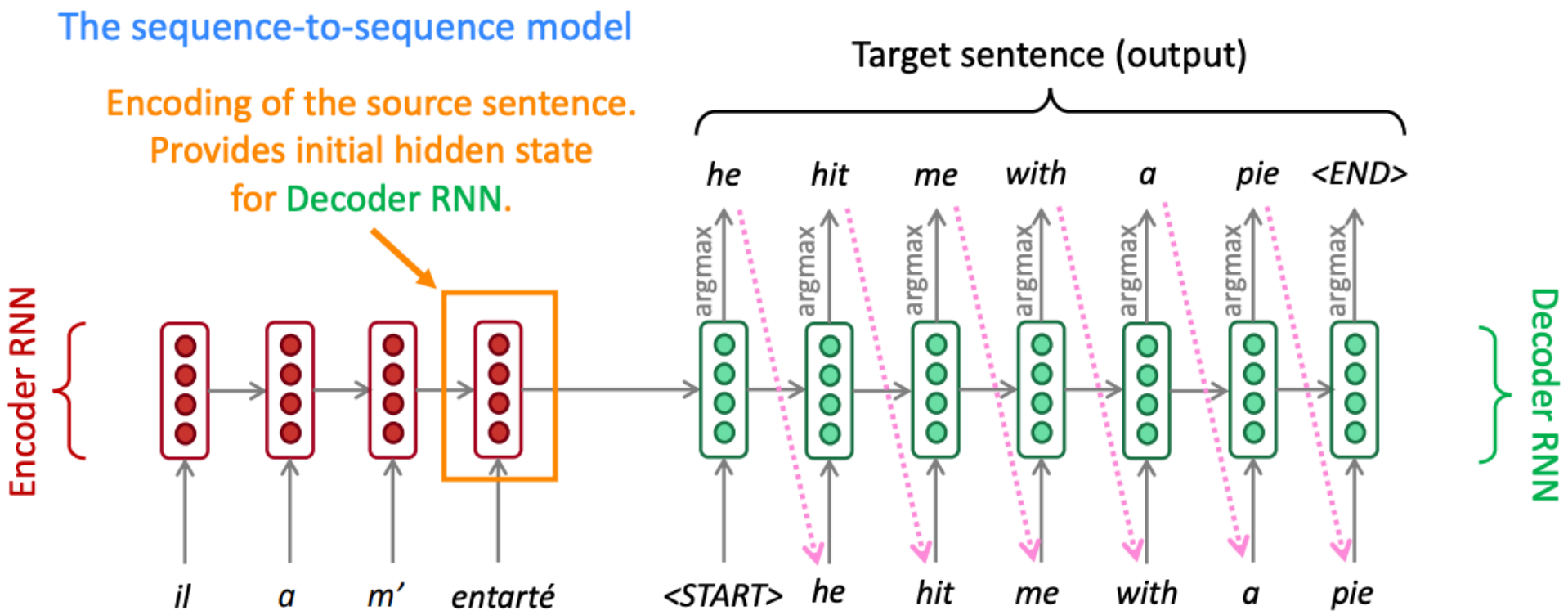
CITY METRO NETWORK EXPANSION WITH REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

Background

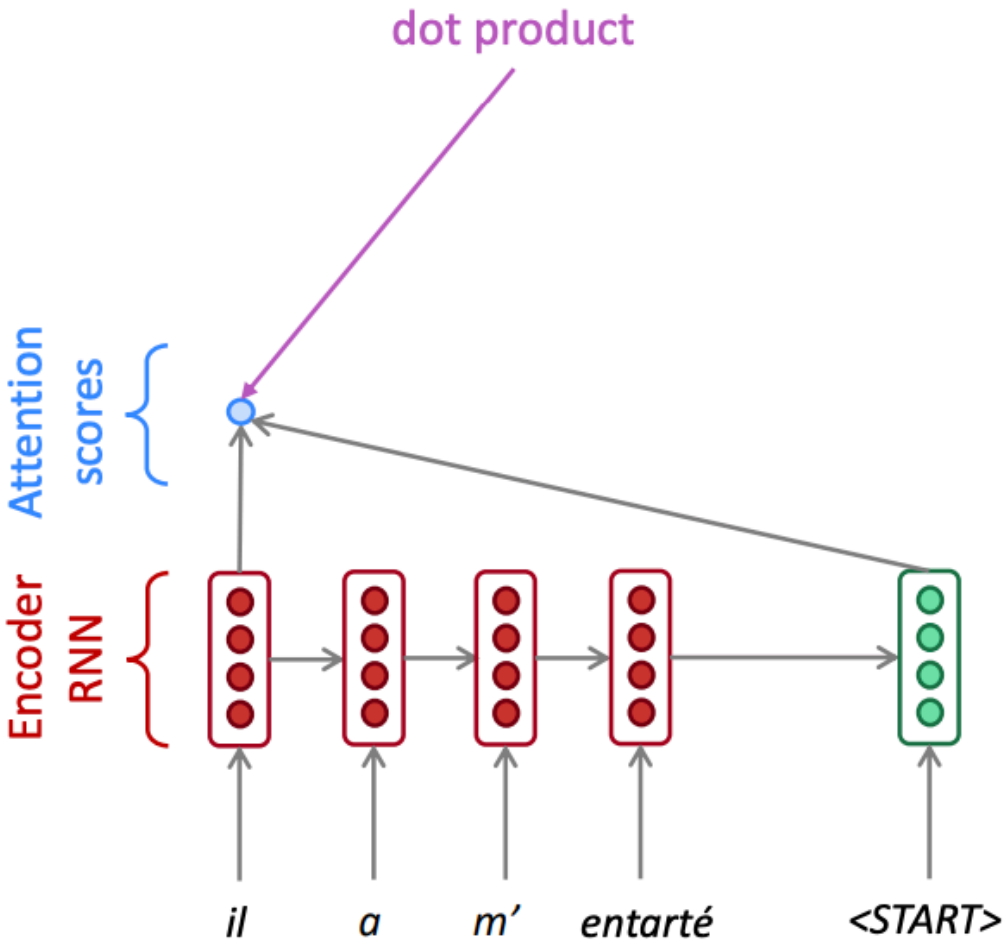
Review: Attention models

Neural machine translation



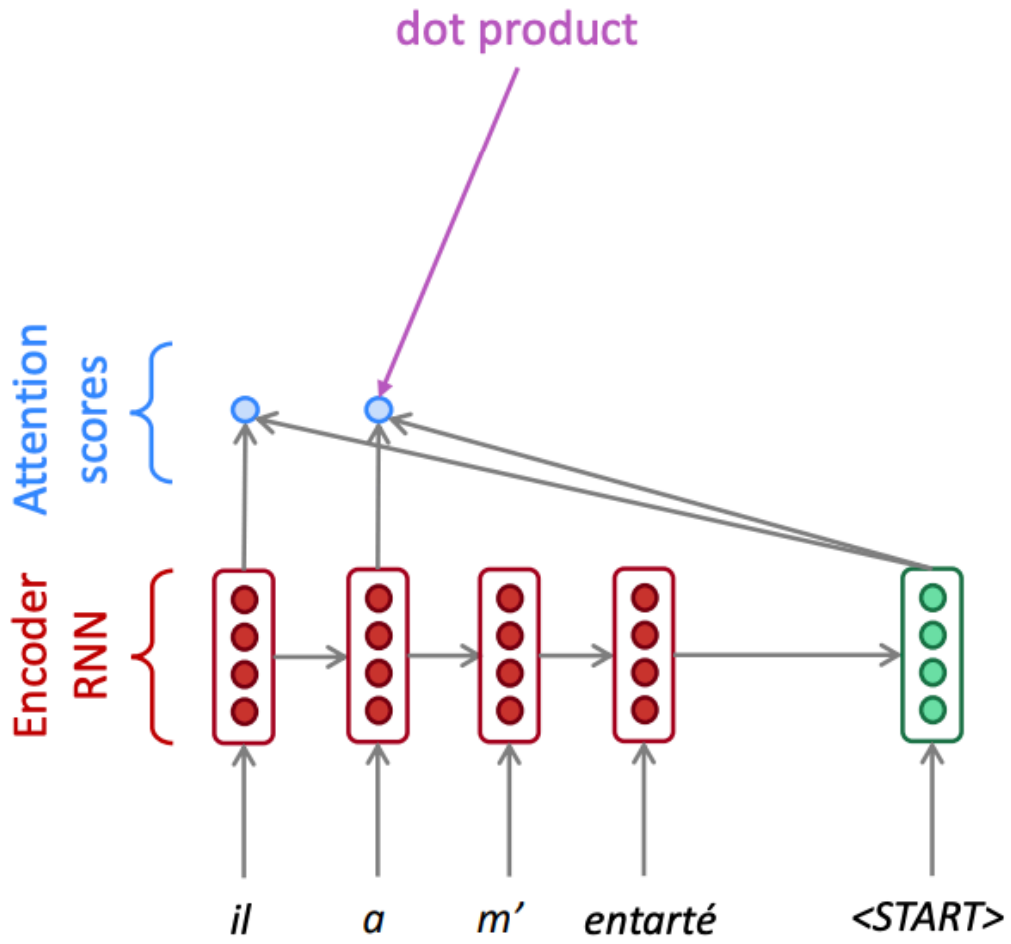
Source: Abigail See

Neural machine translation



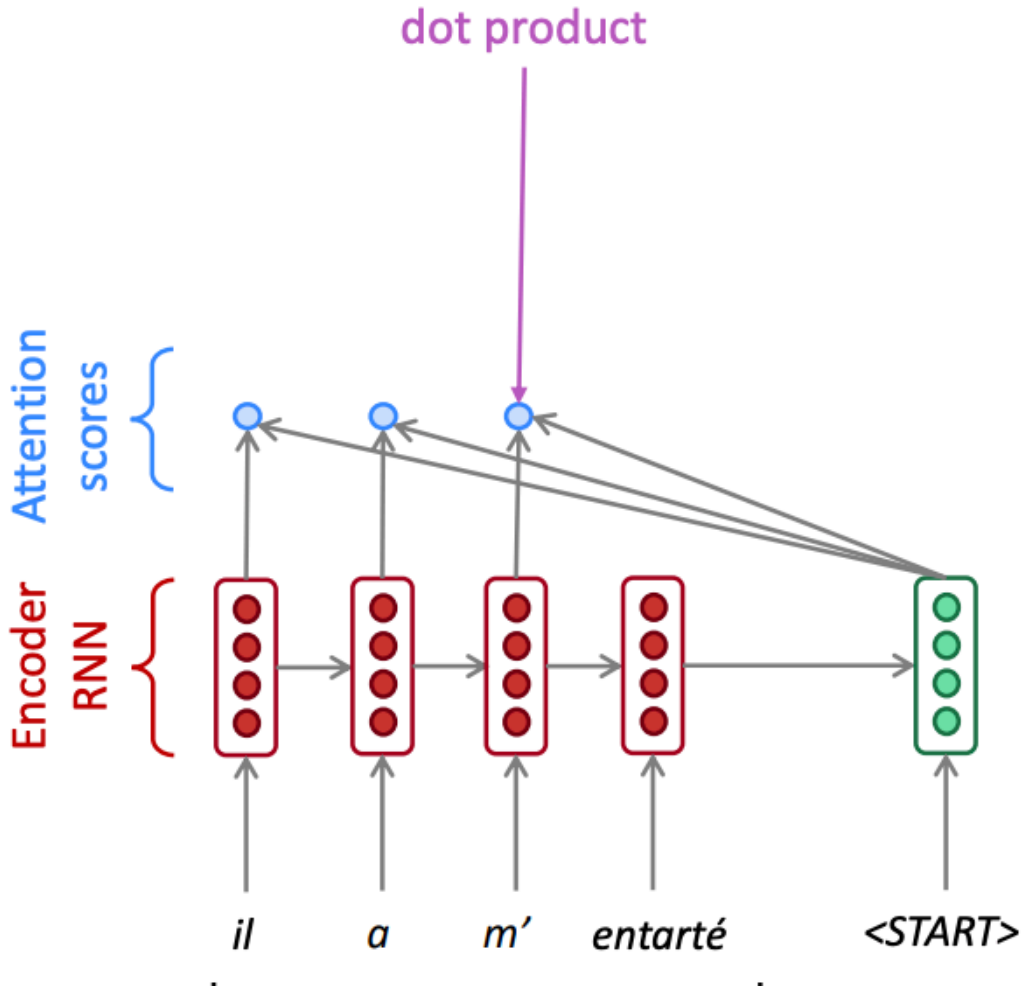
Source: Abigail See

Neural machine translation



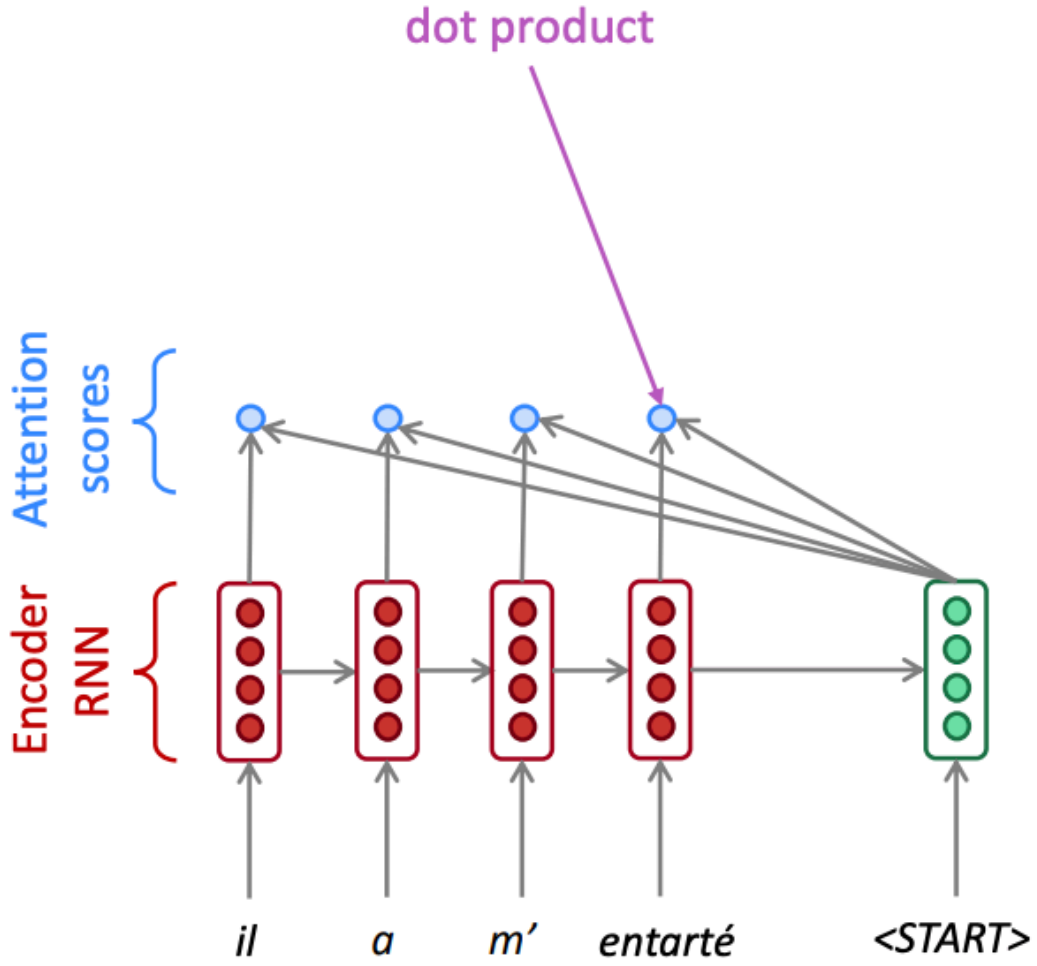
Source: Abigail See

Neural machine translation



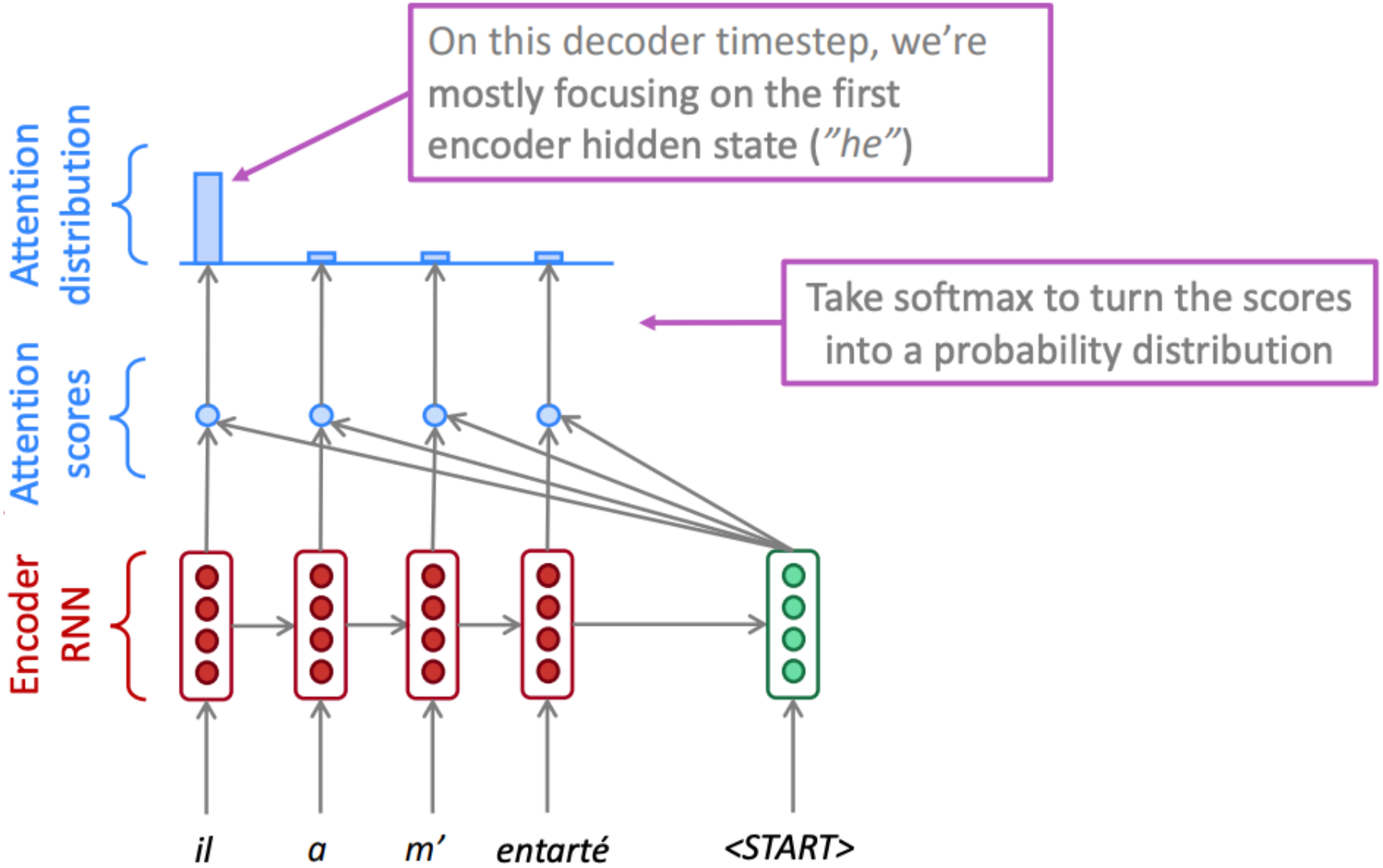
Source: Abigail See

Neural machine translation

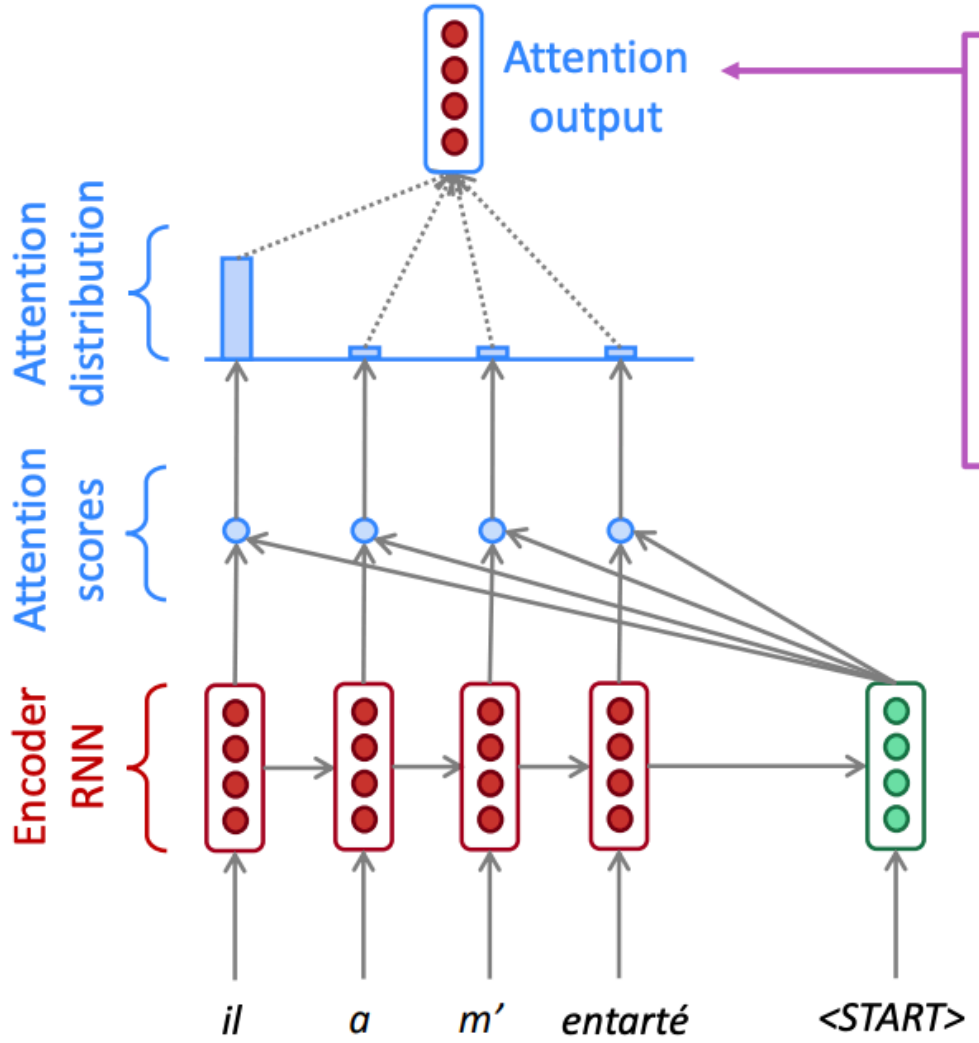


Source: Abigail See

Neural machine translation



Neural machine translation

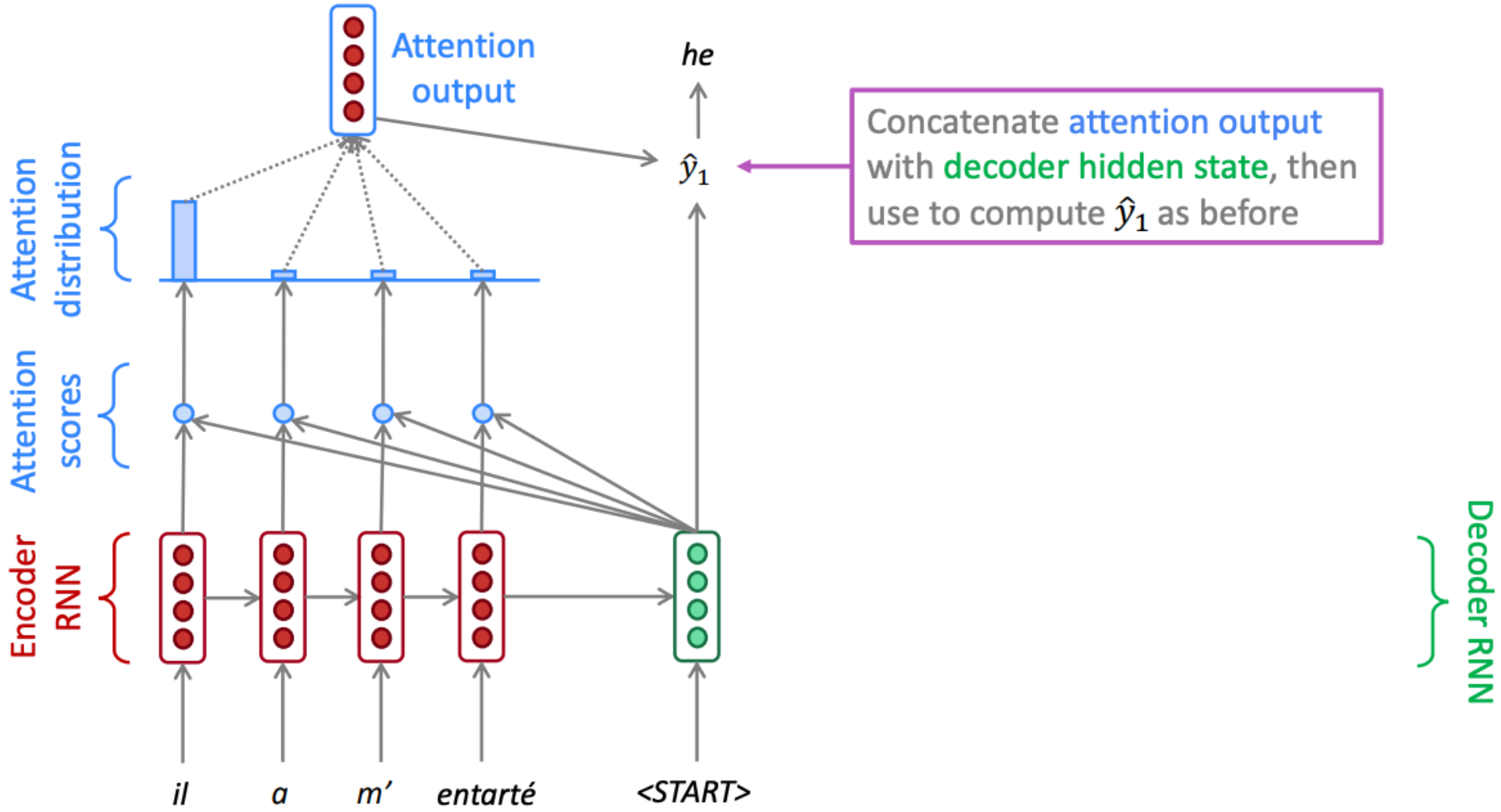


Use the attention distribution to take a **weighted sum** of the encoder hidden states.

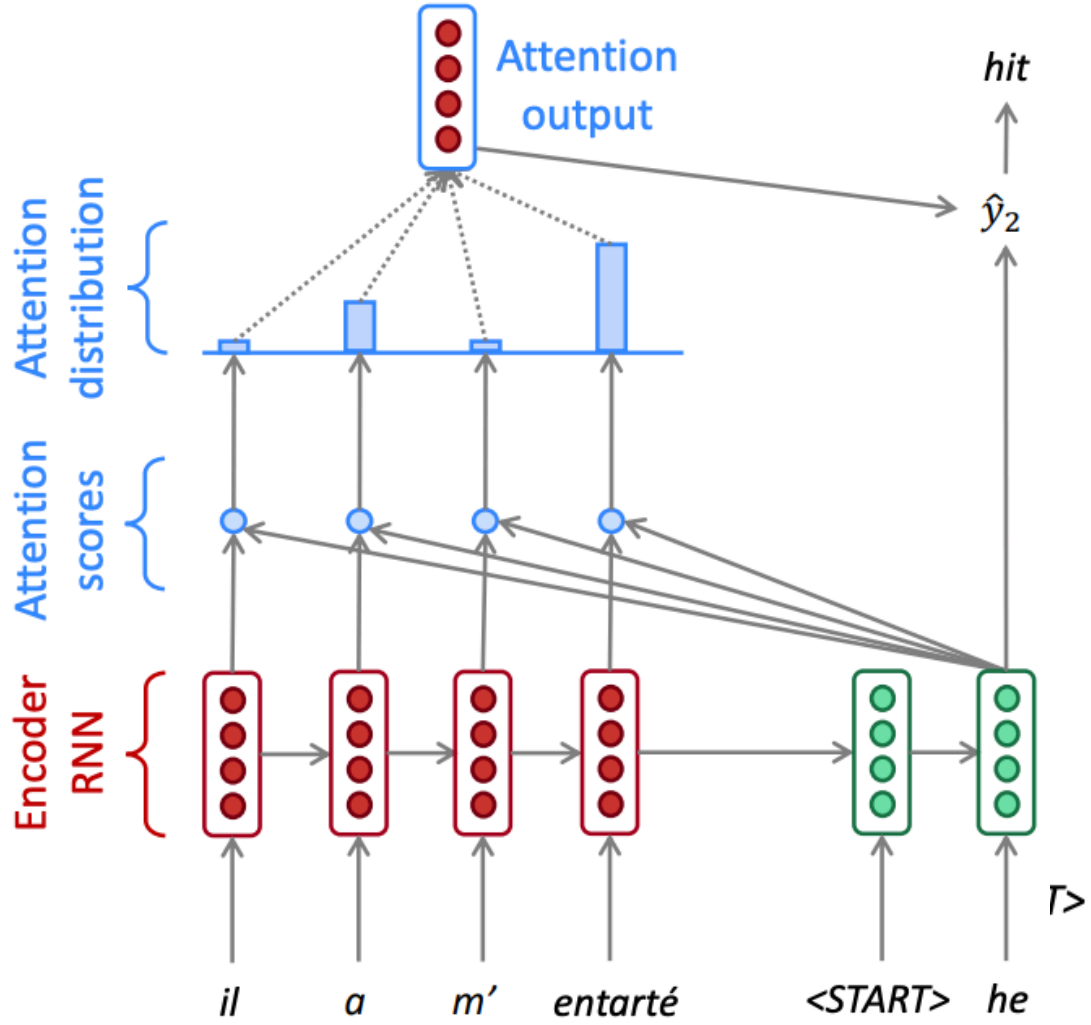
The attention output mostly contains information from the hidden states that received high attention.

Decoder RNN

Neural machine translation

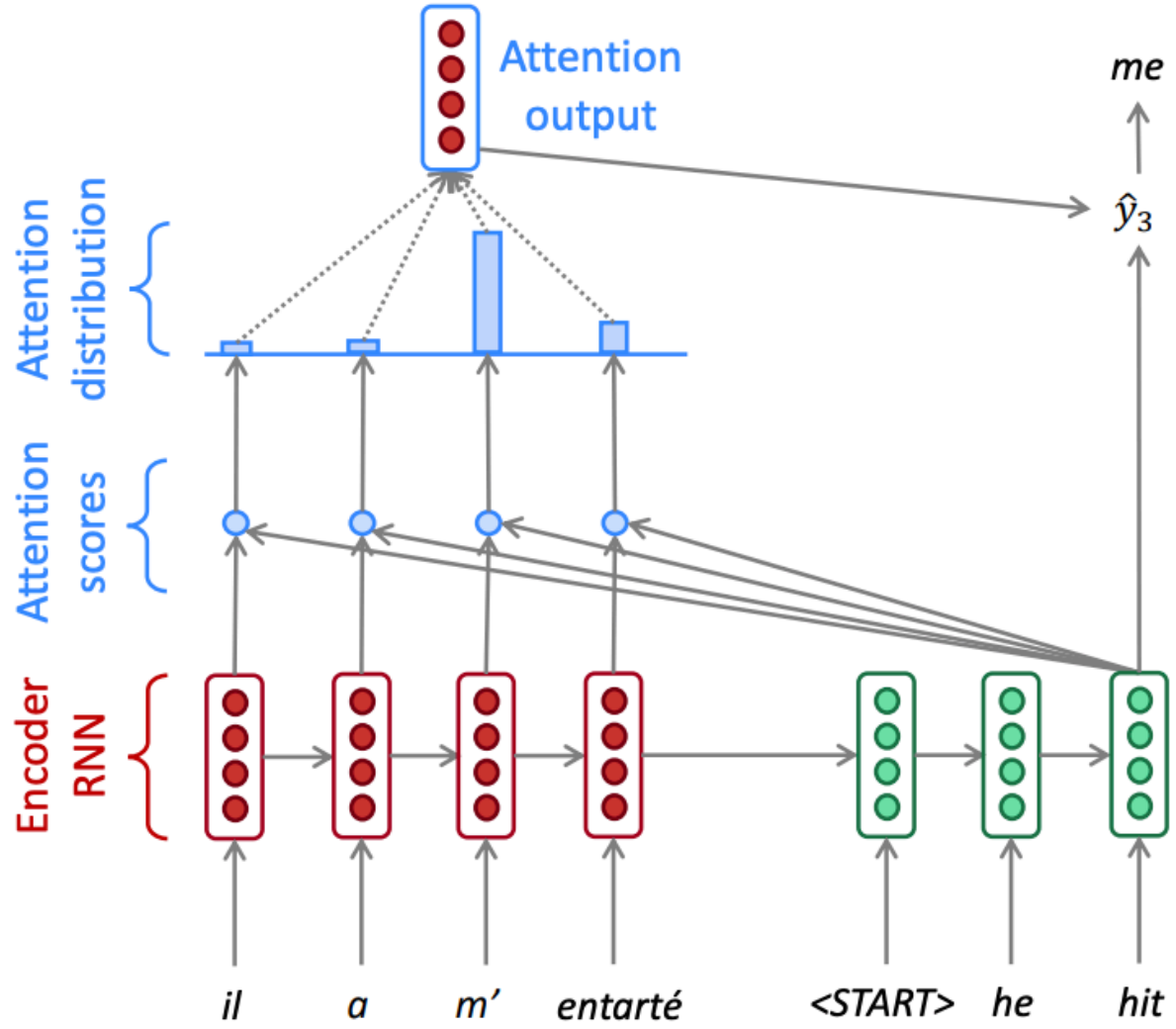


Neural machine translation



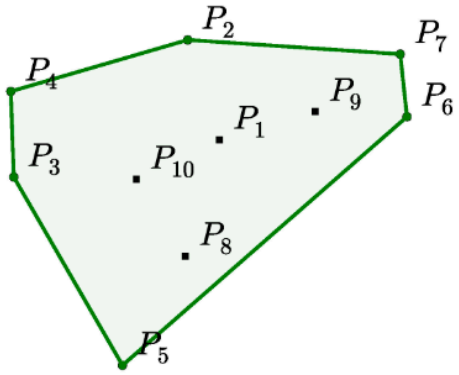
Source: Abigail See

Neural machine translation

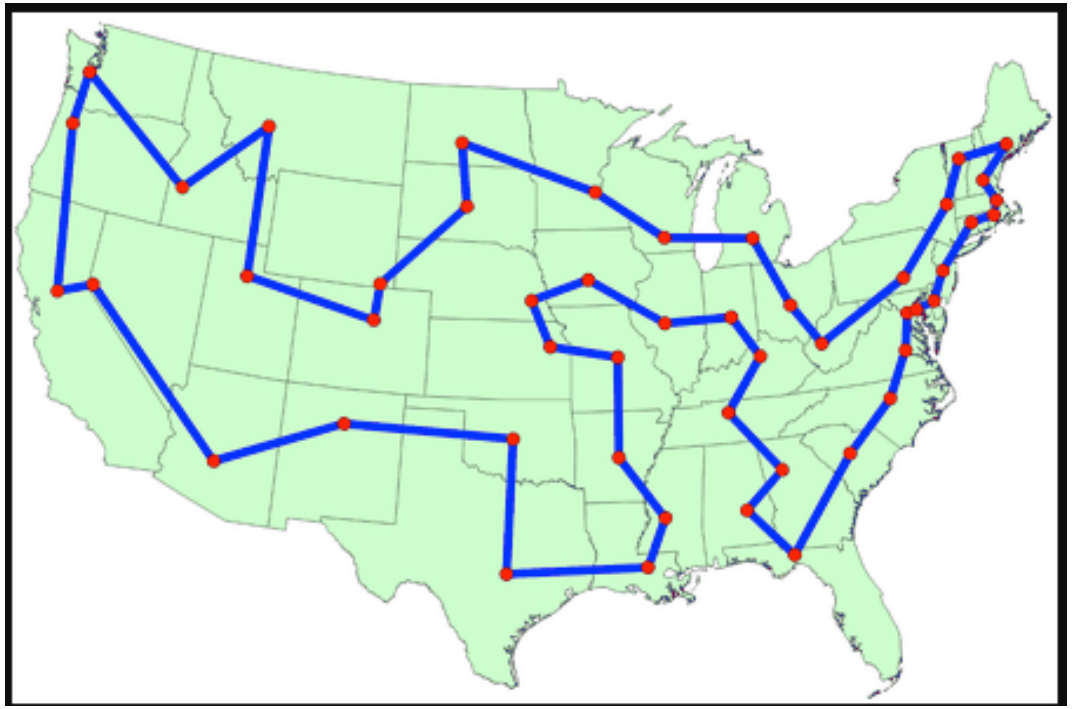


Source: Abigail See

Neural networks for solving combinatorial optimization problems



(a) Input $\mathcal{P} = \{P_1, \ldots, P_{10}\}$, and the output sequence $\mathcal{C}^{\mathcal{P}} = \{\Rightarrow, 2, 4, 3, 5, 6, 7, 2, \Leftarrow\}$ representing its convex hull.

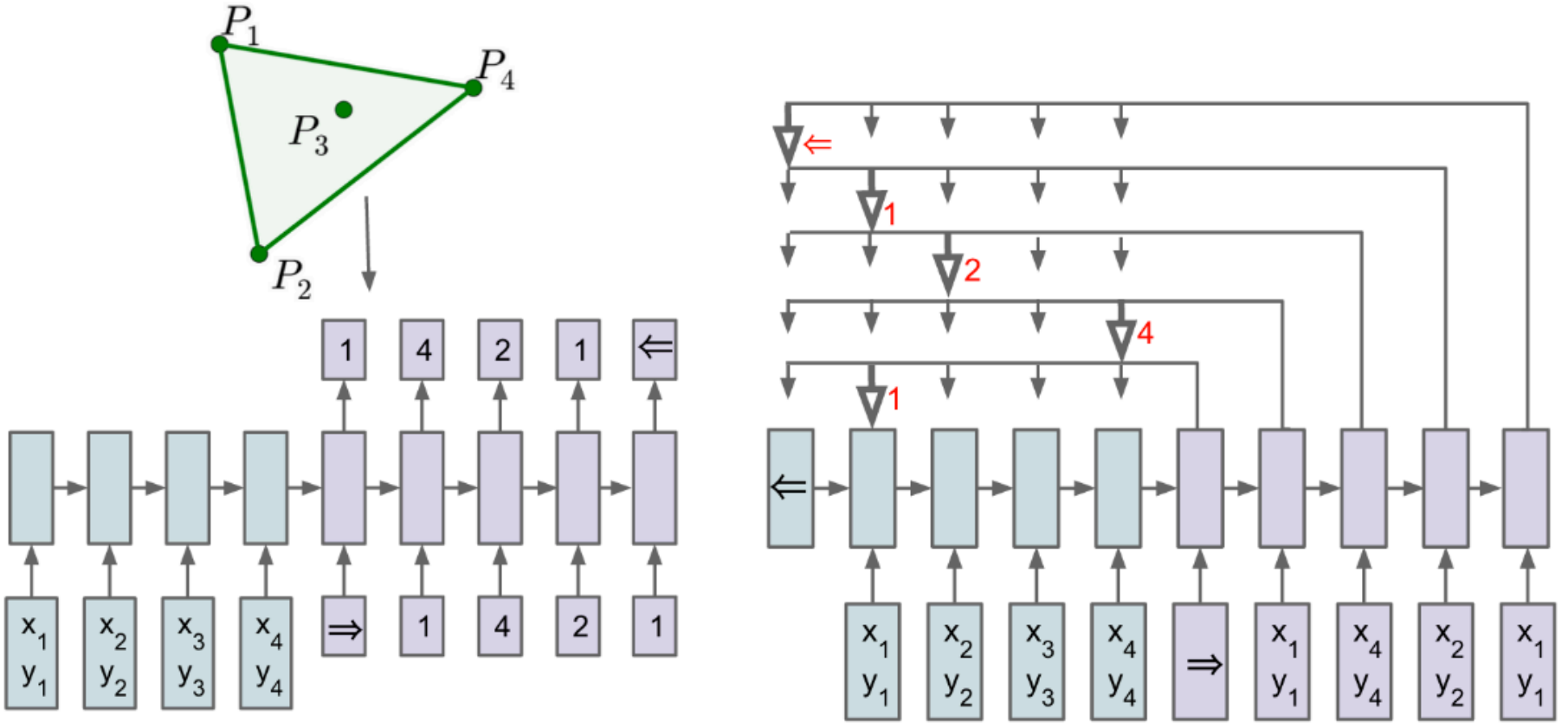


Pointer networks

- Can handle Variable length input

Pointer Networks

Oriol Vinyals* Google Brain
Meire Fortunato* Department of Mathematics, UC Berkeley
Navdeep Jaitly Google Brain



(a) Sequence-to-Sequence

(b) Ptr-Net

- Pointer network is trained in a supervised learning fashion
- Labels are collected from solving problems using typical OR software
- Learning from examples in such a way is undesirable for NP-hard problems because
  - The performance of the model is tied to the quality of the supervised labels
  - Combinatorial -> exponentially growing number of examples needed
  - one cares more about finding a competitive solution more than replicating the results of another algorithm.
- Frame the problem as a sequential decision making one, use policy gradient to estimate the gradient and train the model

NEURAL COMBINATORIAL OPTIMIZATION WITH REINFORCEMENT LEARNING

Irwan Bello*, **Hieu Pham***, **Quoc V. Le, Mohammad Norouzi, Samy Bengio**
Google Brain
{ibello,hyhieu,qvl,mnorouzi,bengio}@google.com

Algorithm 1 Actor-critic training

procedure TRAIN(training set S, number of training steps T, batch size B)
    Initialize pointer network params θ
    Initialize critic network params θ_v
    for $t = 1$ to T **do**
        $s_i \sim \text{SAMPLEINPUT}(S)$ for $i \in \{1, \ldots, B\}$
        $\pi_i \sim \text{SAMPLESOLUTION}(p_\theta(.|s_i))$ for $i \in \{1, \ldots, B\}$
        $b_i \leftarrow b_{\theta_v}(s_i)$ for $i \in \{1, \ldots, B\}$
        $g_\theta \leftarrow \frac{1}{B} \sum_{i=1}^{B} (L(\pi_i|s_i) - b_i) \nabla_\theta \log p_\theta(\pi_i|s_i)$
        $\mathcal{L}_v \leftarrow \frac{1}{B} \sum_{i=1}^{B} \|b_i - L(\pi_i)\|_2^2$
        $\theta \leftarrow \text{ADAM}(\theta, g_\theta)$
        $\theta_v \leftarrow \text{ADAM}(\theta_v, \nabla_{\theta_v} \mathcal{L}_v)$
    end for
    return θ
end procedure

CITY METRO NETWORK EXPANSION WITH REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

Motivation

- Existing studies formulate the problem as non- linear integer programming

- The huge solution space makes it difficult to find a good solution effectively
  - One common method in existing studies is to limit the search space
  - predefines corridors based on expert knowledge, and only consider to design metro lines in these corridors.

- Carefully handcrafted heuristics, embedded with problem-specific knowledge, are usually efficient for large search space
  - However, the factors considered in the expansion of metro line vary with different cities and stages

- Method without expert knowledge to solve the city metro network expansion problem using RL

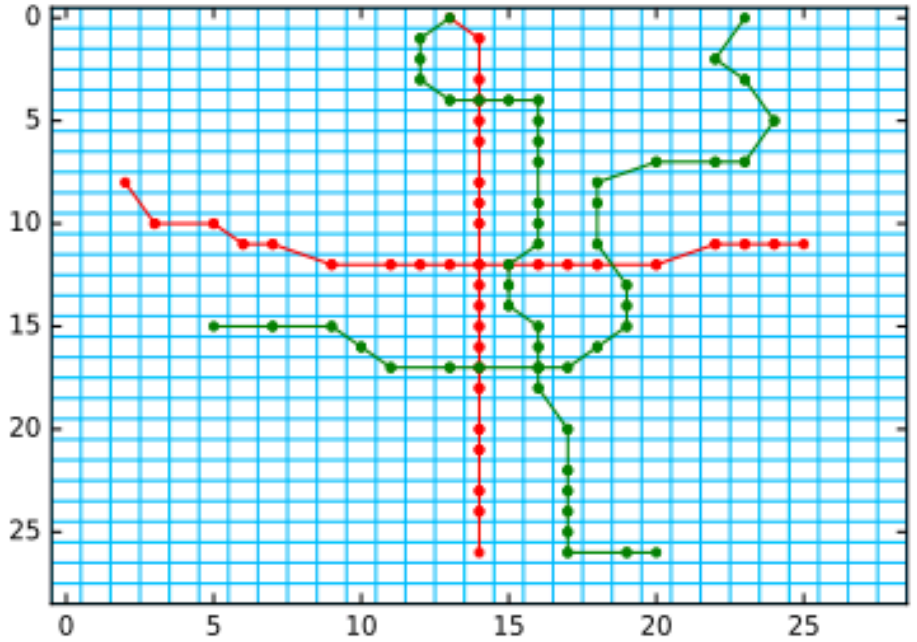
- Formulate the metro line expansion as a process of sequential station selection, an MDP, and design feasibility rules based on the selected station sequence to ensure the reasonable connection patterns of the metro line.

- Based on real city-scale human mobility information of 24,770,715 mobile phone users obtained from a citywide 3G cellular network in Xi’an, China

- Incorporate social equity concerns into metro network expansion

Problem formulation

- N by N grid (width d)
- Each centroid: candidate station
- Each grid has a compound index of development
- OD demand is symmetrical



(a) Metro network.

Constraints

- The consecutive stations must follow the min-max distance rules (?)
- Line routing should avoid subtour and meandering lines
- Number of stations is limited by N
- The budget is limited by B
- For a new line Z, the objective (reward) is defined as :

$$w(Z) = \alpha_1 \times R_{od}(Z) + \alpha_2 \times R_{ac}(Z)$$

$$\alpha_1 + \alpha_2 = 1$$

- Station selected at step t z_t
- Selected station sequence up to time t $Z_t = \{z_1, z_2, \ldots, z_t\}$
- Given graph G,

$$P(Z|G) = \prod_{t=1}^{T} P(z_t|G, Z_{t-1})$$

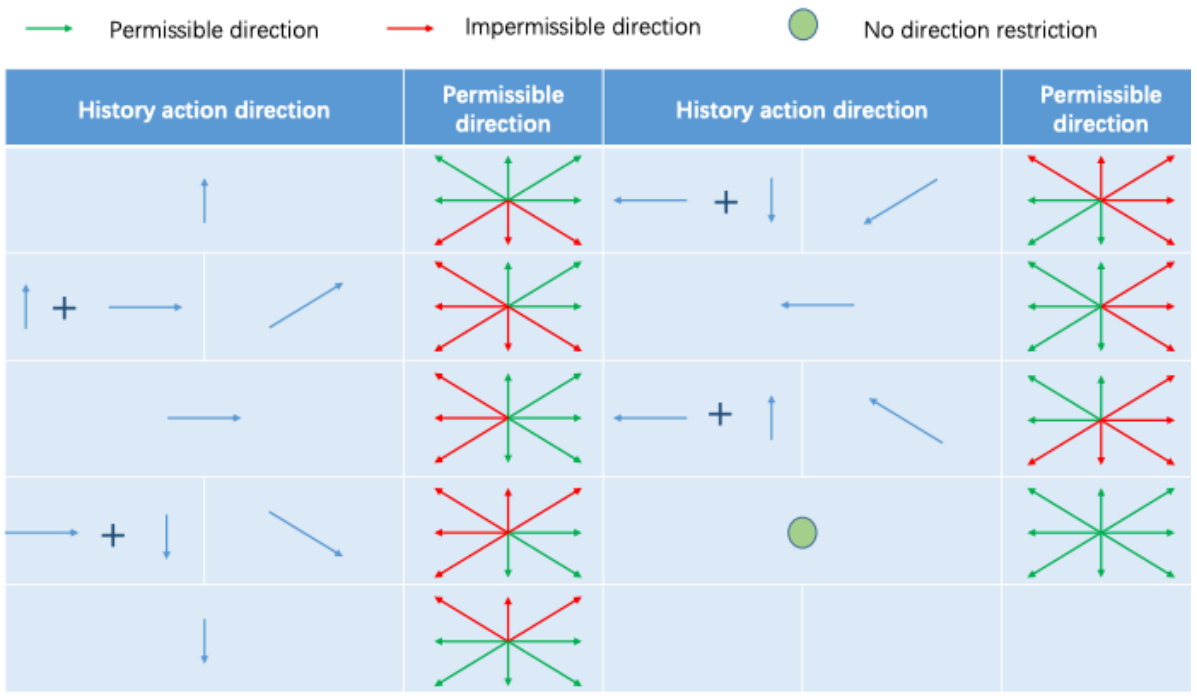
$$M_t = M(Z_t)$$

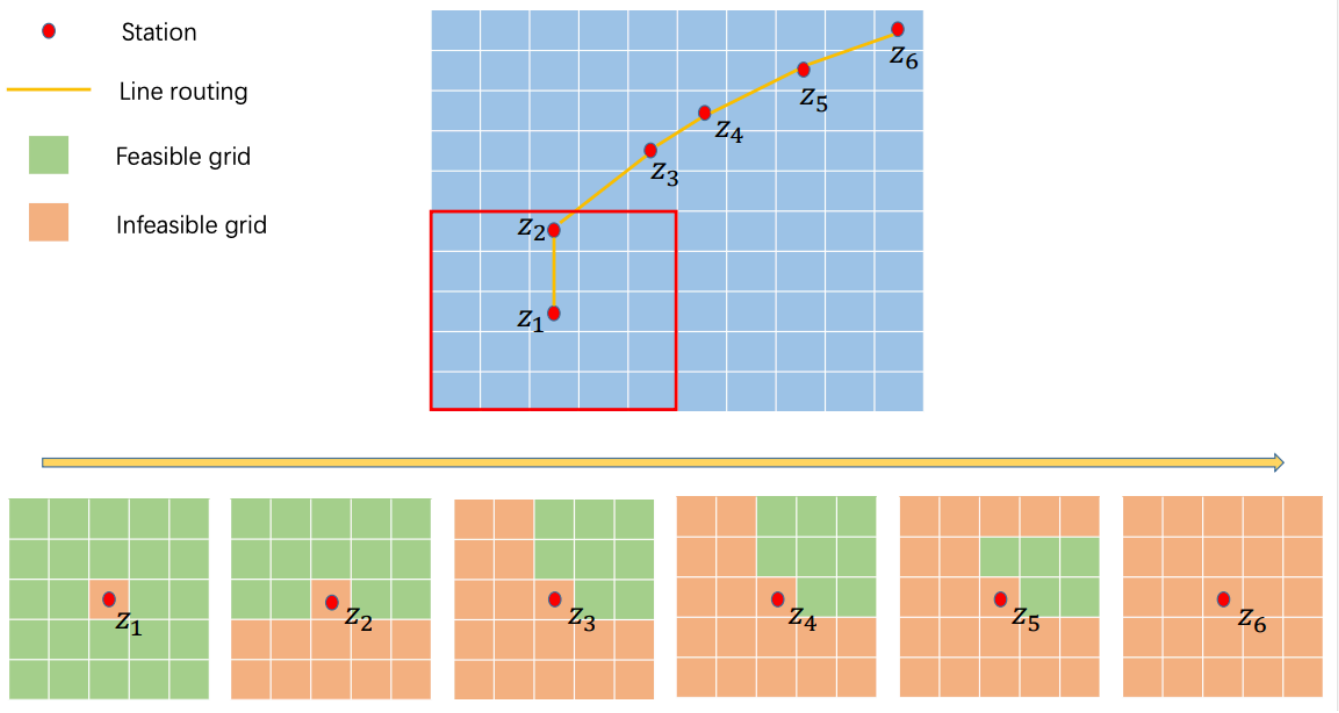
- Binary vector indicating feasibility rules

MDP formulation

- State : $Z_t = \{z_1, z_2, \ldots, z_t\}$
- Action : z_t
- Reward : $w(Z) = \alpha_1 \times R_{od}(Z) + \alpha_2 \times R_{ac}(Z)$

Feasibility rules





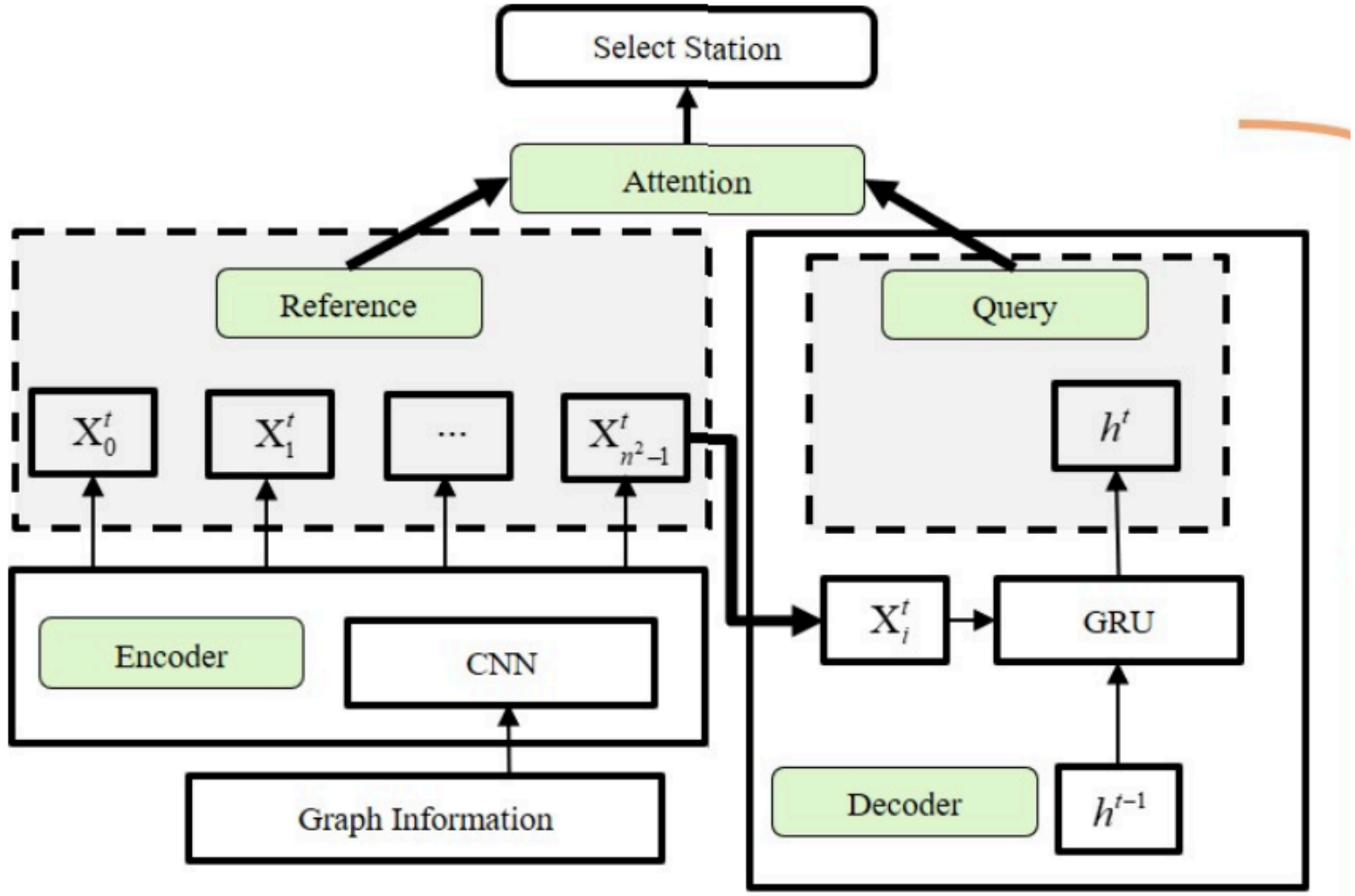
Network architecture

- Seq2seq model

$$q_i^t = v_a^T \tanh(W_a[X_i^t; h^t]),\ a^t = \mathrm{softmax}(q^t)$$

$$c^t = \sum_i a_i^t X_i^t$$

$$P(z_{t+1}|\mathcal{G}, Z_t) = \mathrm{softmax}(v_c^T \tanh(W_c[X_i^t; c^t]) \oplus H \cdot M_t)$$



Algorithm 1 Processing Procedure

Input Graph $\mathcal{G}$

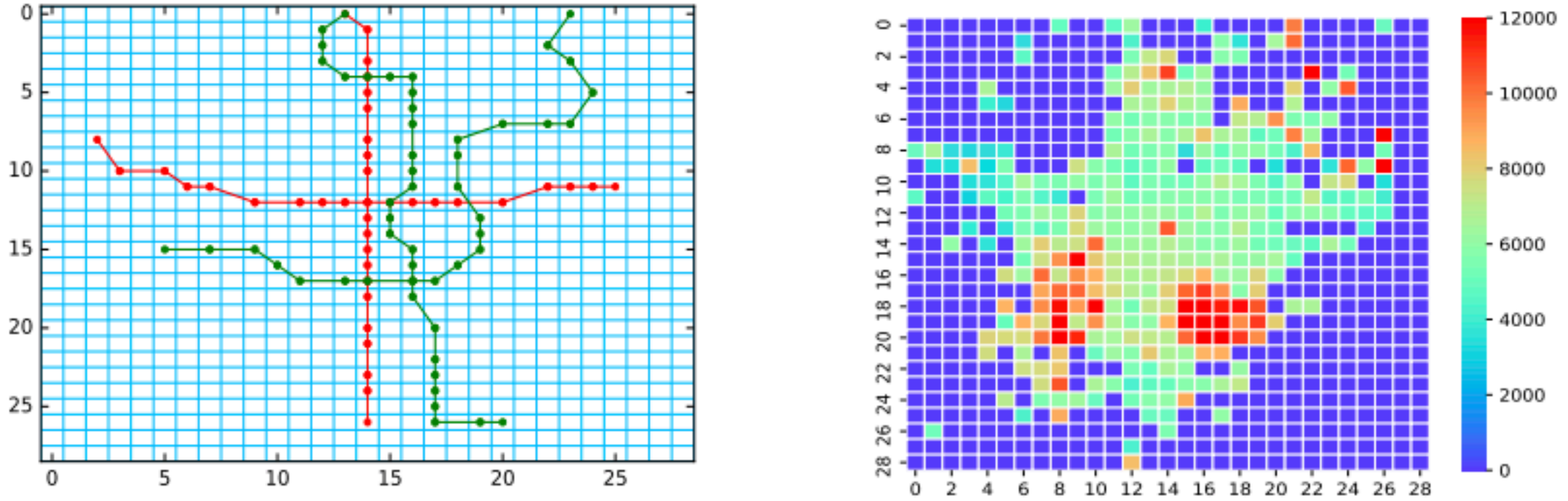
for $\mathrm{t} = 0, ..., \mathrm{T} - 1$ **do**
    Update the feasible stations M_t by Equation (3).
    Compute embeddings X^t of the current state by encoder.
    Compute the query h^t according to last hidden state h^{t-1} and the embedding of last selected station X_i^t by decoder (Initialize h^0 when $t = 0$).
    Compute the attention value a^t by Equation (4), and the probability $P(z_{t+1}|\mathcal{G}, Z_t)$ by Equation (6).
    Select Station z_{t+1} with the probability $P(z_{t+1}|\mathcal{G}, Z_t)$.
end for
return Solution $= Z_T$

Algorithm 2 Actor Critic Training

Require: Batch size B, Training epoch E, Step T

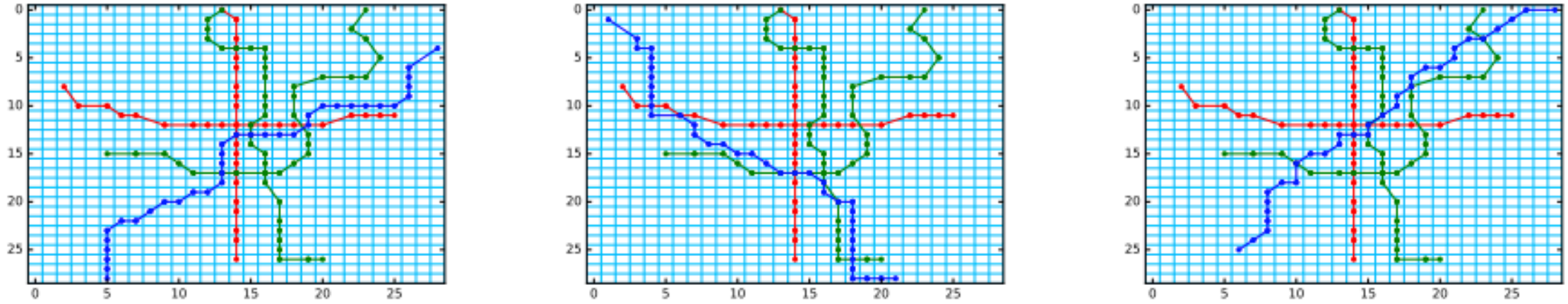
Initialize actor parameters θ
Initialize critic parameters θ_c
for epoch $= 1, ..., \mathrm{E}$ **do**
    Reset city metro network $\mathcal{G}$
    for $\mathrm{t} = 1, ..., \mathrm{T}$ **do**
        Select the next station by actor, $z_t \leftarrow \mathrm{SampleSolution}\ (p(\cdot|\theta))$
        Update feasibility rules $M_t = M(Z_t)$
    end for
    Find a solution Z^i for each batch, where $i \in \{1, \ldots, B\}$
    Calculate baseline $b(\mathcal{G})$ by critic
    $\nabla J(\theta|\mathcal{G}) \leftarrow \frac{1}{B} \sum_{i=1}^{B} (\omega(Z^i|\mathcal{G}) - b(\mathcal{G})) \nabla_\theta \log p_\theta \left(Z^i|\mathcal{G}\right)$
    Update the parameters of actor $\theta \leftarrow \mathrm{Adam}(\theta, \nabla J(\theta|\mathcal{G}))$
    $\nabla L_c \leftarrow \frac{1}{B} \sum_{i=1}^{B} (\omega(Z^i|\mathcal{G}) - b(\mathcal{G}))^2$
    Update the parameters of critic $\theta_c \leftarrow \mathrm{Adam}(\theta_c, \nabla L_c)$
end for

Case study



(a) Metro network. (b) House price (RMB).

Figure 3: The current city operational status.



(a) $\alpha_1 = 1, \alpha_2 = 0$ (b) $\alpha_1 = 0.5, \alpha_2 = 0.5$ (c) $\alpha_1 = 0, \alpha_2 = 1$

Figure 4: The next expanded metro line with higher priority. The blue lines are our expanding lines.

Table 1: The newly satisfied OD trips and social equity

	R_{od}	R_{ac}	ω
$\alpha_1, \alpha_2 = 1, 0$	48.17	34.60	48.17
$\alpha_1, \alpha_2 = 0.5, 0.5$	26.79	33.18	29.98
$\alpha_1, \alpha_2 = 0, 1$	27.38	36.64	36.64

Reinforcement Learning for Solving the Vehicle Routing Problem

Mohammadreza Nazari
Department of Industrial Engineering
Lehigh University
Bethlehem, PA 18015
mon314@lehigh.edu

Afshin Oroojlooy
Department of Industrial Engineering
Lehigh University
Bethlehem, PA 18015
afo214@lehigh.edu

Martin Takáč
Department of Industrial Engineering
Lehigh University
Bethlehem, PA 18015
takac@lehigh.edu

Lawrence V. Snyder
Department of Industrial Engineering
Lehigh University
Bethlehem, PA 18015
lvs2@lehigh.edu

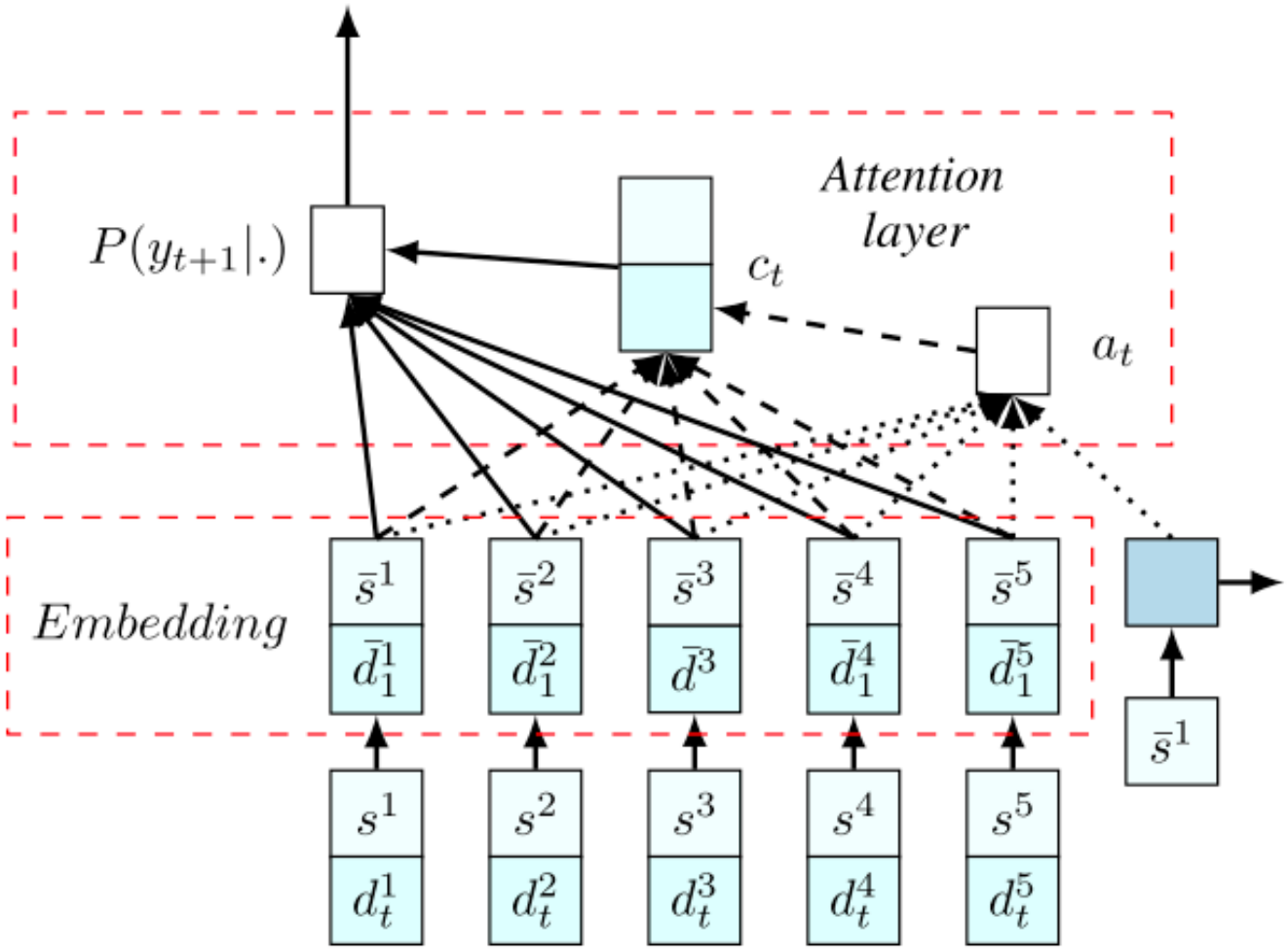
Problem definition

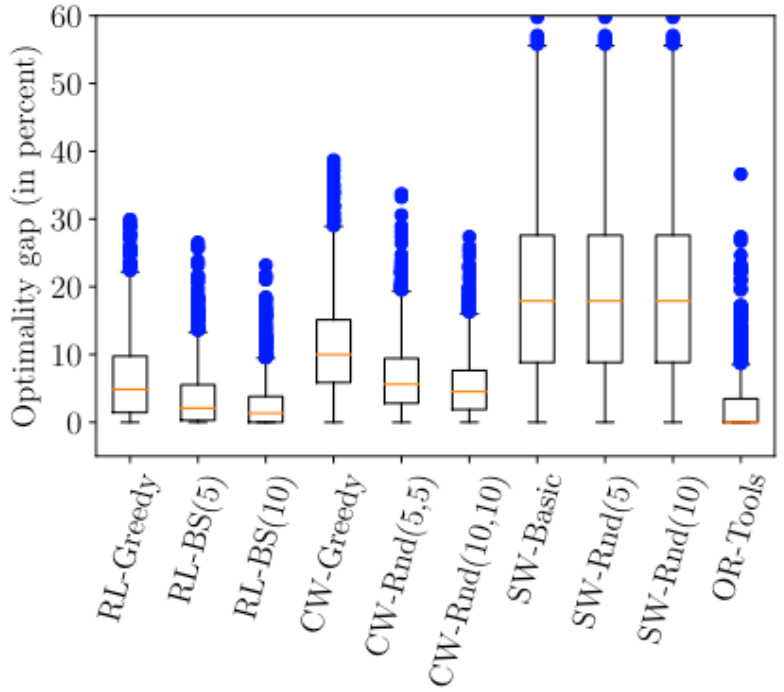
- A single capacitated vehicle is responsible for delivering items to multiple customer nodes
  - It’s really just solving TSP
- The vehicle must return to the depot to pick up additional items when it runs out.
- All trips starting and ending at a single node (depot)
- Reward: negative average distance

- Using classical heuristics for VRP, the entire distance matrix must be recalculated and the system must be re-optimized from scratch, which is often impractical, especially if the problem size is large.

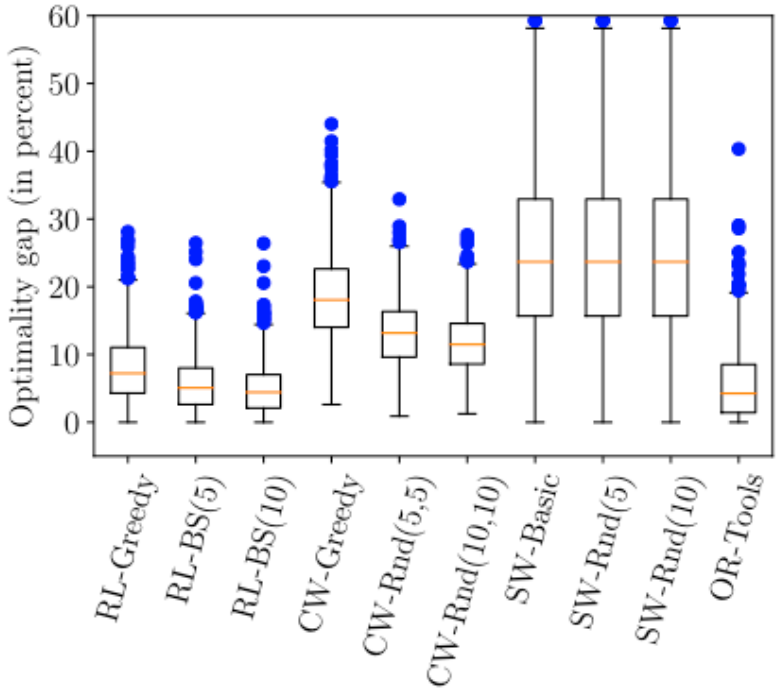
- In contrast, this framework does not require an explicit distance matrix, and only one feed-forward pass of the network will update the routes based on the new data.

Network architecture





(a) Comparison for VRP10



(b) Comparison for VRP20

	RL-Greedy	RL-BS(5)	RL-BS(10)	CW-Greedy	CW-Rnd(5,5)	CW-Rnd(10,10)	SW-Basic	SW-Rnd(5)	SW-Rnd(10)	OR-Tools
RL-Greedy		12.2	7.2	99.4	97.2	96.3	97.9	97.9	97.9	41.5
RL-BS(5)	85.8		12.5	99.7	99.0	98.7	99.1	99.1	99.1	54.6
RL-BS(10)	91.9	57.7		99.8	99.4	99.2	99.3	99.3	99.3	60.2
CW-Greedy	0.6	0.3	0.2		0.0	0.0	68.9	68.9	68.9	1.0
CW-Rnd(5,5)	2.8	1.0	0.6	92.2		30.4	84.5	84.5	84.5	3.5
CW-Rnd(10,10)	3.7	1.3	0.8	97.5	68.0		86.8	86.8	86.8	4.7
SW-Basic	2.1	0.9	0.7	31.1	15.5	13.2		0.0	0.0	1.4
SW-Rnd(5)	2.1	0.9	0.7	31.1	15.5	13.2	0.0		0.0	1.4
SW-Rnd(10)	2.1	0.9	0.7	31.1	15.5	13.2	0.0	0.0		1.4
OR-Tools	58.5	45.4	39.8	99.0	96.5	95.3	98.6	98.6	98.6	

(c) Comparison for VRP50

	RL-Greedy	RL-BS(5)	RL-BS(10)	CW-Greedy	CW-Rnd(5,5)	CW-Rnd(10,10)	SW-Basic	SW-Rnd(5)	SW-Rnd(10)	OR-Tools
RL-Greedy		25.4	20.8	99.9	99.8	99.7	99.5	99.5	99.5	44.4
RL-BS(5)	74.4		35.3	100.0	100.0	99.9	100.0	100.0	100.0	56.6
RL-BS(10)	79.2	61.6		100.0	100.0	100.0	99.8	99.8	99.8	62.2
CW-Greedy	0.1	0.0	0.0		0.0	0.0	65.2	65.2	65.2	0.0
CW-Rnd(5,5)	0.2	0.0	0.0	92.6		32.7	82.0	82.0	82.0	0.7
CW-Rnd(10,10)	0.3	0.1	0.0	97.2	65.8		85.4	85.4	85.4	0.8
SW-Basic	0.5	0.0	0.2	34.8	18.0	14.6		0.0	0.0	0.0
SW-Rnd(5)	0.5	0.0	0.2	34.8	18.0	14.6	0.0		0.0	0.0
SW-Rnd(10)	0.5	0.0	0.2	34.8	18.0	14.6	0.0	0.0		0.0
OR-Tools	55.6	43.4	37.8	100.0	99.3	99.2	100.0	100.0	100.0	

(d) Comparison for VRP100

Policy gradient vs value function approximation

- Pros
  - Effective in high-dimensional action space
  - Easy to use in continuous action space
  - Can learn stochastic policies
- Cons
  - High variance
  - Costly policy evaluation
  - Often converge to a local optimum

That’s it for me!