Shunting Trains with Deep Reinforcement Learning

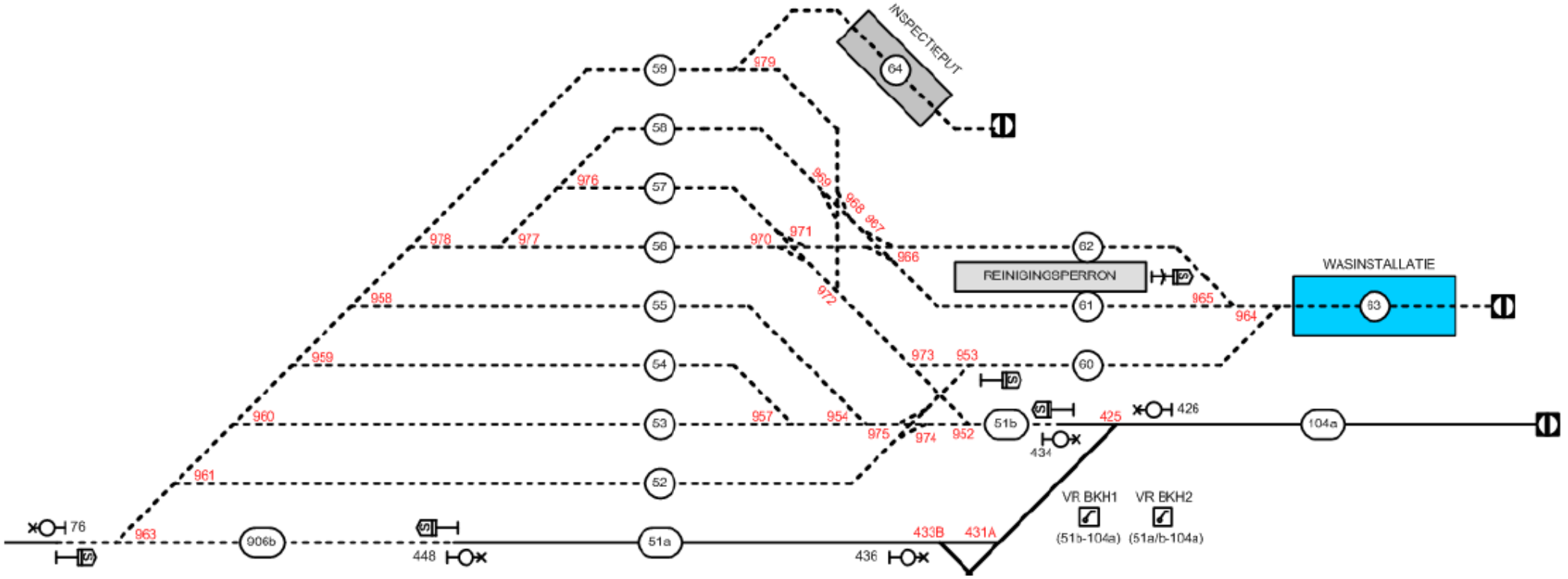
Wan-Jui Lee

R&D Hub Logistics,

Dutch Railways

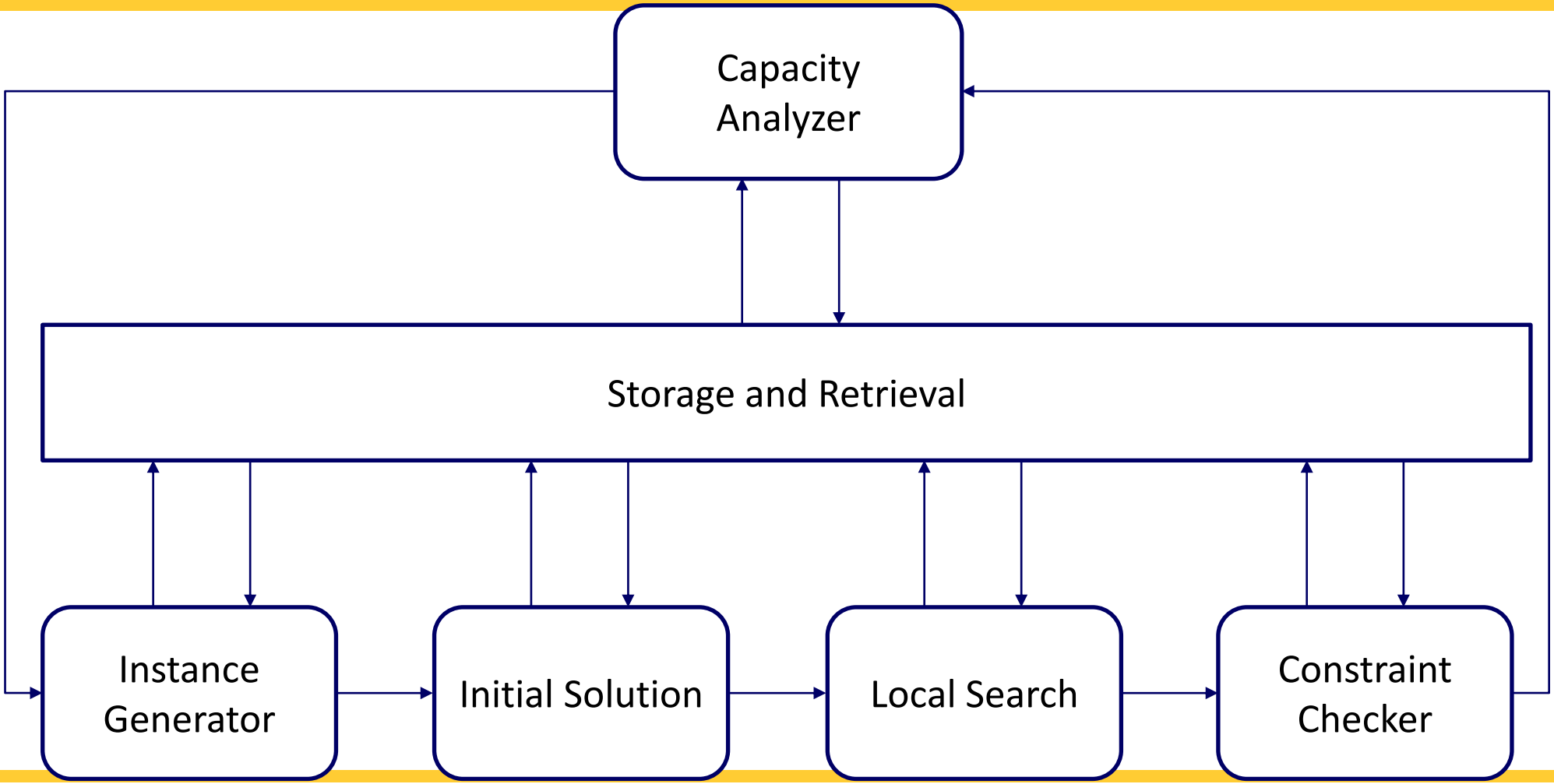
Meet the fleet of NS

Train Unit Shunting Problem



Service Location with carousel layout (Den Haag Kleine Binckhorst)

Shunt Plan Simulator



Example of a Shunting Instance

In

Zoeken:

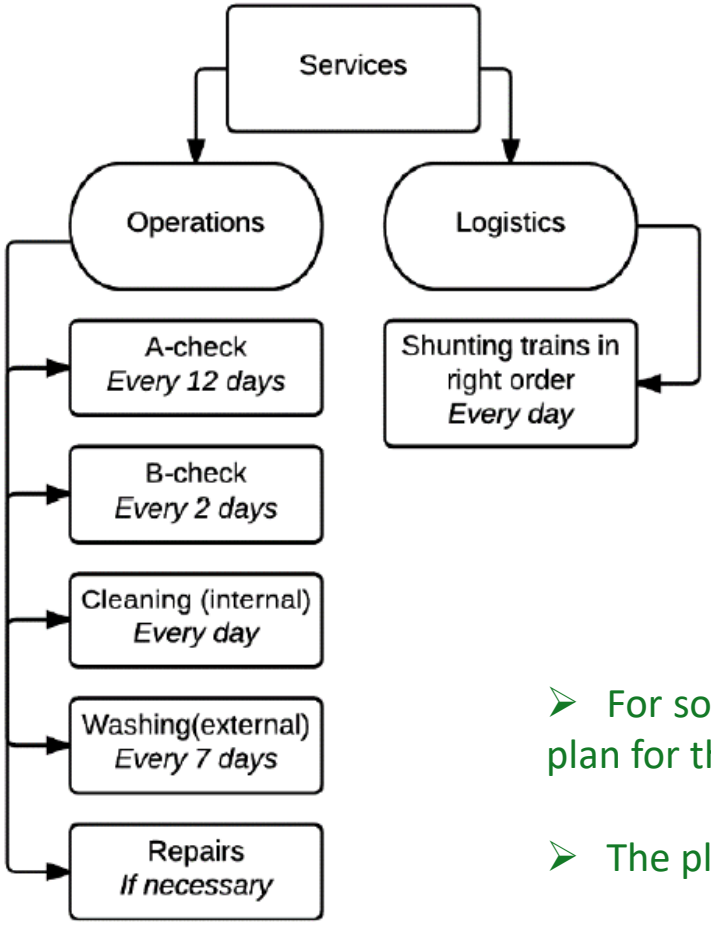
Naam	Aankomsttijd	Treinstellen	Taken	Spoornummer
1000	08:45	2401	BCO	Spoor906a
2000	09:13	8608; 9401	BCO	Spoor906a
3000	09:40	7501	ZER	Spoor104a
4000	10:13	9402	BCO	Spoor906a
5000	10:38	9404; 9403	BCO	Spoor104a
6000	10:54	2601	ZER; BCO	Spoor104a
7000	18:56	2402	BIR; BCO	Spoor906a
8000	19:12	9405	BIR	Spoor906a
9000	20:18	2403; 2602	BIR; BCO	Spoor906a
10000	20:34	8610	BCO	Spoor906a
11000	21:44	7608	BIR; BCO	Spoor906a
12000	22:33	8614; 9406	BIR; BCO	Spoor906a
13000	23:12	8615	BIR; ACO	Spoor906a
14000	23:21	9407; 8621	BIR; BCO	Spoor906a
15000	23:28	2603; 2405; :	BIR; ACO; BC	Spoor906a
16000	01:03	9411; 9409	BIR; BCO	Spoor906a
17000	01:21	9412	BIR; BCO	Spoor906a
18000	01:35	2407; 2406; :	BIR; BCO	Spoor906a

Uit

Zoeken:

Naam	Vertrektijd	Treinstellen	Spoornummer
51000	15:04	24_ _	Spoor906a
52000	16:47	94_ _	Spoor906a
53000	14:34	86_ _	Spoor906a
54000	14:20	75_ _	Spoor906a
55000	16:04	*9402	Spoor906a
56000	13:36	*9404; *9403	Spoor104a
57000	14:36	26_ _	Spoor104a
58000	22:58	*2402	Spoor906a
59000	05:18	*9405	Spoor906a
60000	04:28	*2403; *2602	Spoor906a
61000	22:20	*8610	Spoor906a
62000	05:01	76_ _	Spoor906a
63000	05:59	86_ _; *9406	Spoor906a
64000	06:48	*8615	Spoor906a
65000	05:34	*8621	Spoor104a
66000	05:24	*9407; 94_ _; *9409	Spoor906a
67000	05:44	*2405; *2404	Spoor906a
68000	04:55	*2603	Spoor906a
69000	07:47	*9412	Spoor906a
70000	05:32	24_ _; *2406; 26_ _	Spoor906a

Processes at Service Sites



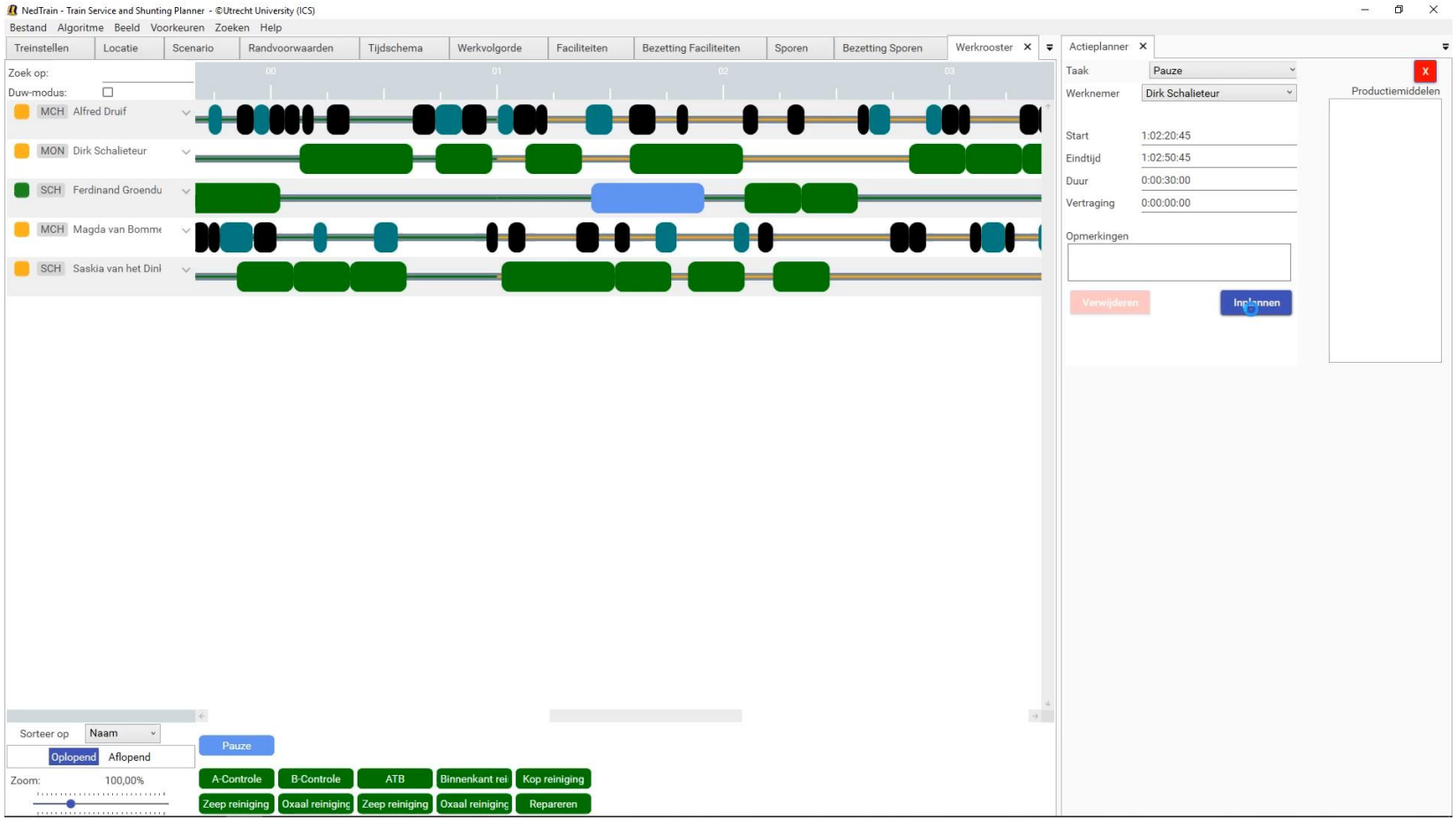
Problems to solve

- Shunting
- Routing
- Parking
- Matching
- Service
- Combine
- Split

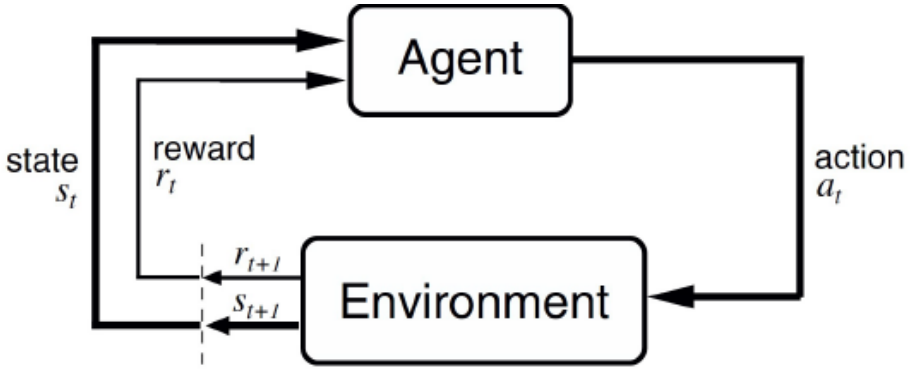
➢ For some yards it takes a human planner up to one day to create a plan for the upcoming night.

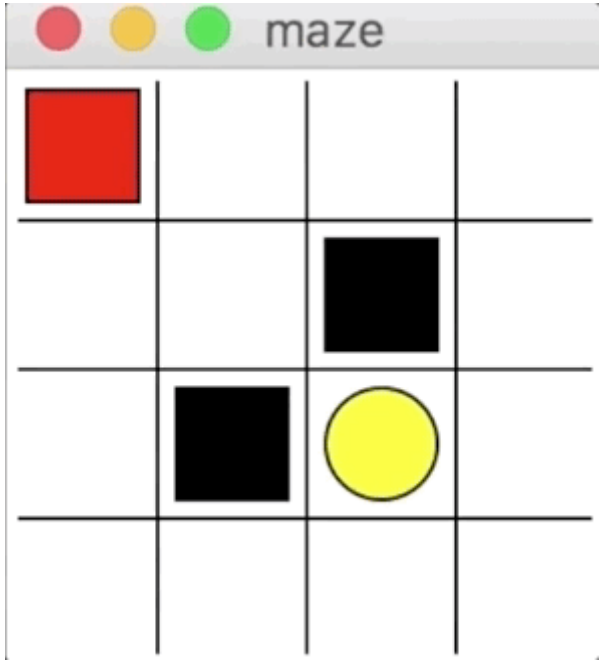
➢ The planning task is getting more complex due to increase of trains

What a Shunting Schedule Looks Like



Can machines learn to plan?



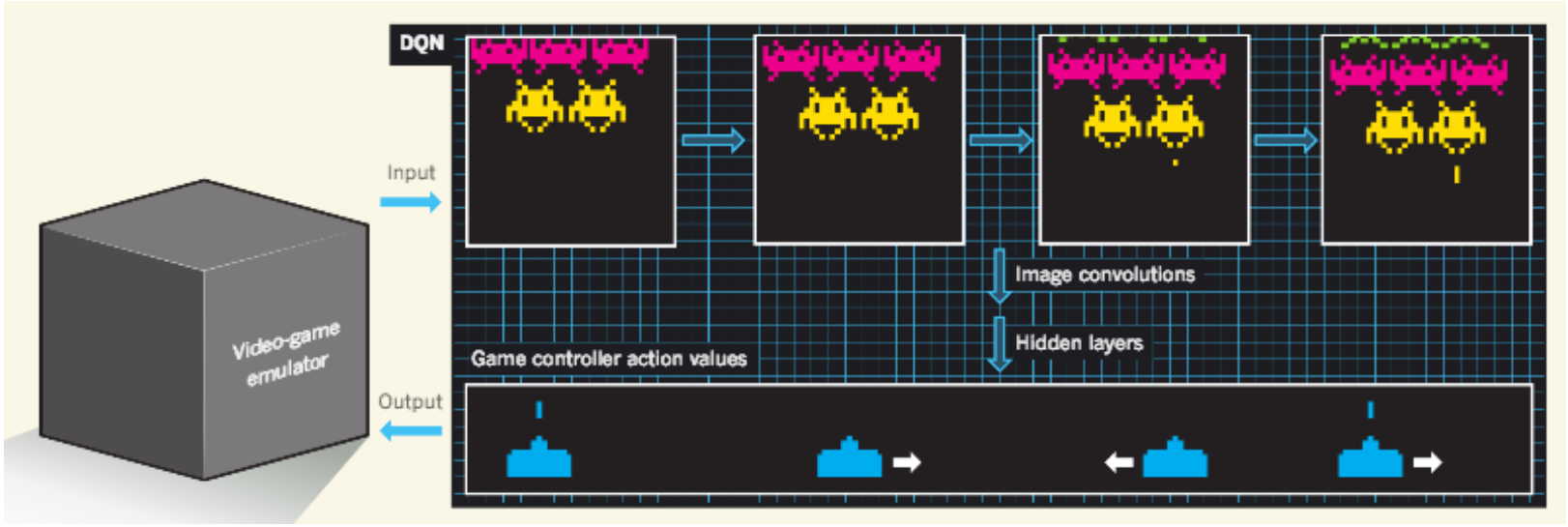


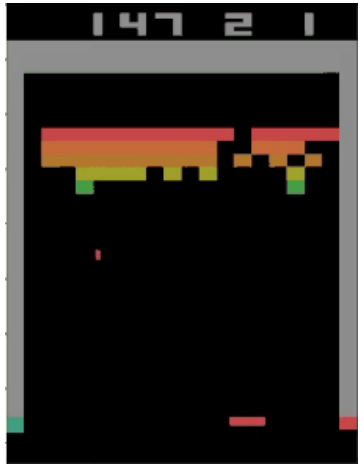
Reinforcement Learning learns to play a game by gaining experience, just like a human player:

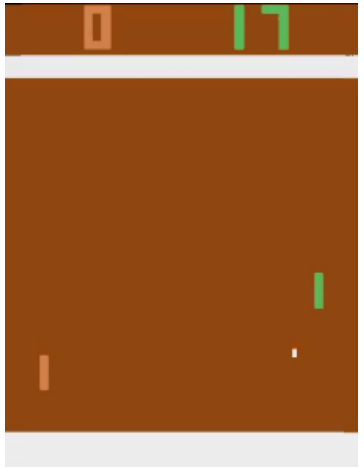
- Try various actions in different situations (explore)
- Learn/store information about the game that can bem

  generalized to potentially unseen scenarios
- Learn the most valuable actions by using the reward signal (exploit)

Deep Q-Network (DQN) by Google Deepmind







Reinforcement Learning + Deep Neural Networks

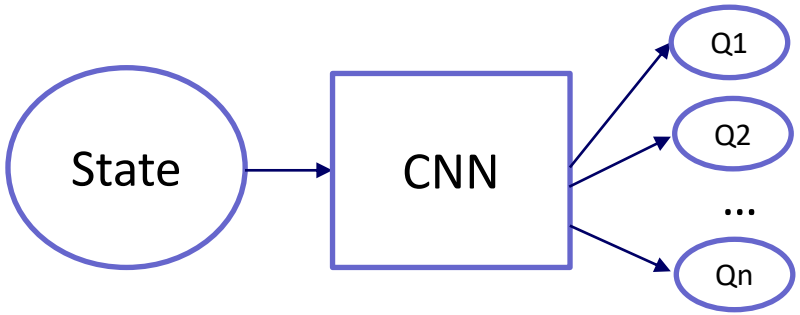
Q-Learning

- A popular Reinforcement Learning algorithm
- An extension to traditional dynamic programming
- It learns the value for each state-action pair: $Q(s; a)$.

Q-learning does not scale: we need to store (and learn) each state-action pair explicitly in the Q-Table.

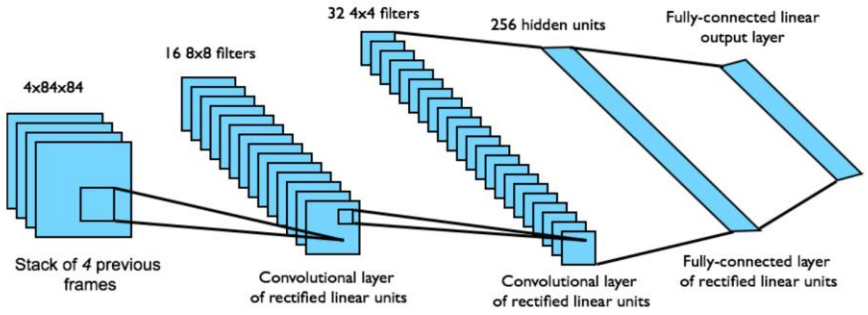
Deep Reinforcement Learning

Deep Q-Network (DQN) of Mnih (2015) represents Q-Table using a Convolutional Neural Network.



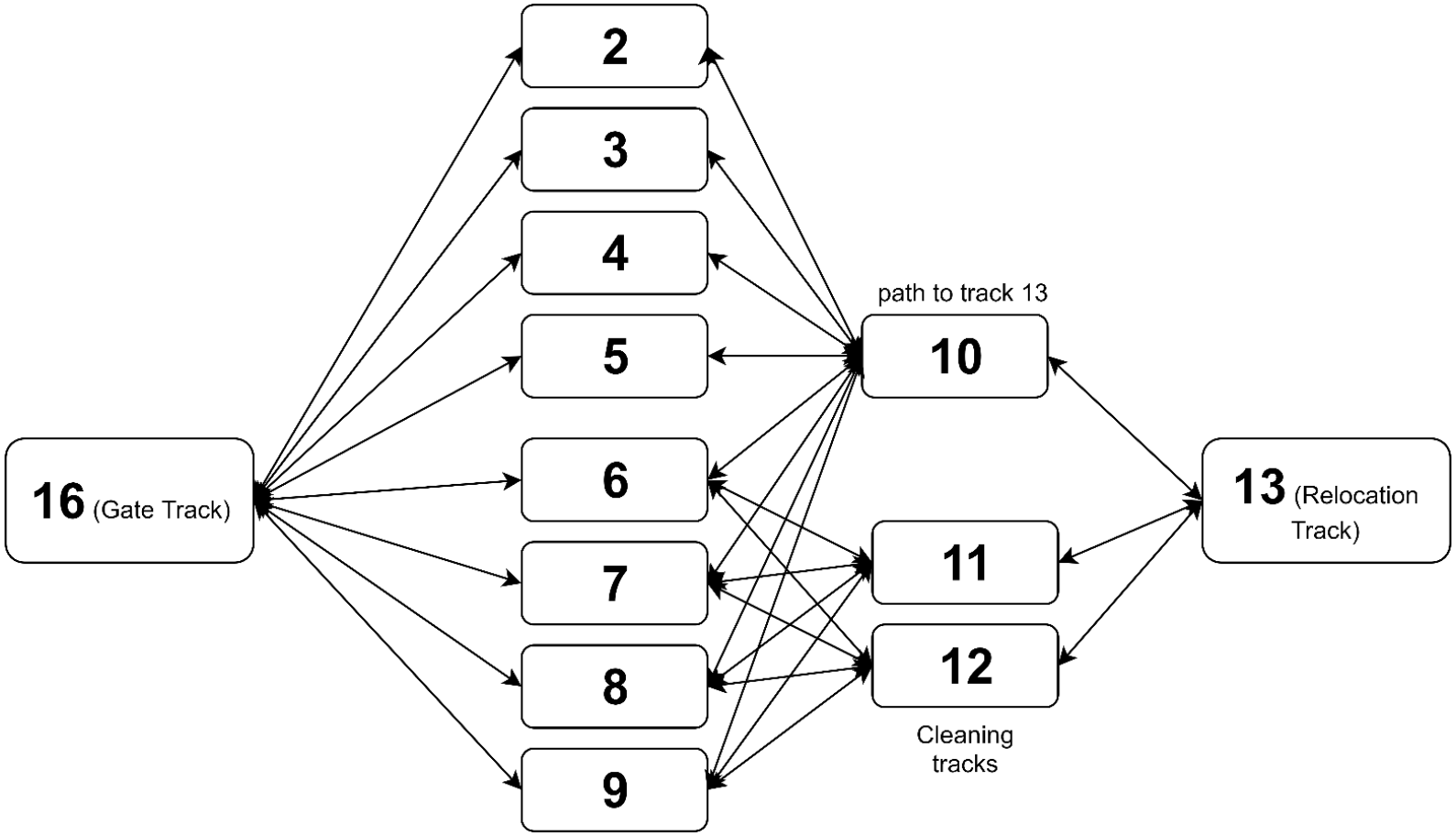
DQN in Atari

- End-to-end learning of values $Q(s, a)$ from pixels s
- Input state s is stack of raw pixels from last 4 frames
- Output is $Q(s, a)$ for 18 joystick/button positions
- Reward is change in score for that step



- Combine reinforcement learning with Deep Neural Networks
- No need to learn all state-action pairs explicitly

DRL for TUSP including Service Tasks



Scope of the Problem to be Solved

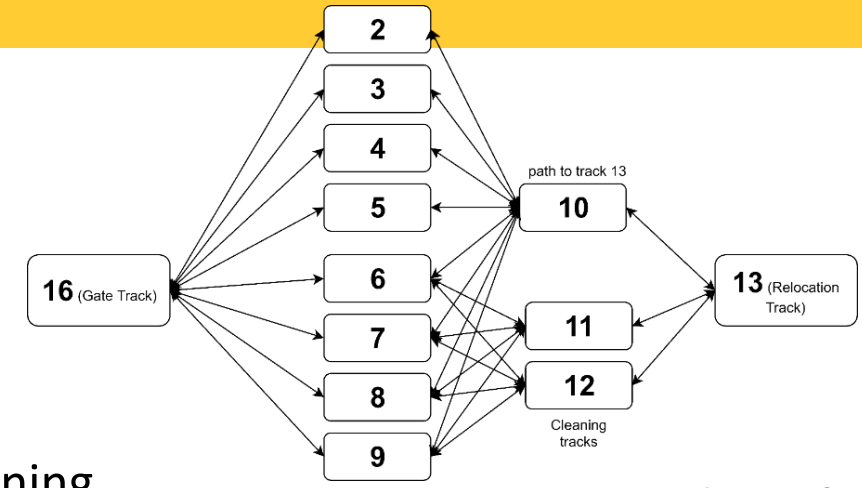
- Single unit trains both in arrival and departure
- Cleaning service
- Cleaning starts as soon as train put on cleaning track
- No simultaneous movement
- Agent can move trains as much as time budget allows
- Full information on schedule of trains
- Trains arrival and task time deterministic
- Trains must leave exactly on time

State Space Design (Input to NN)

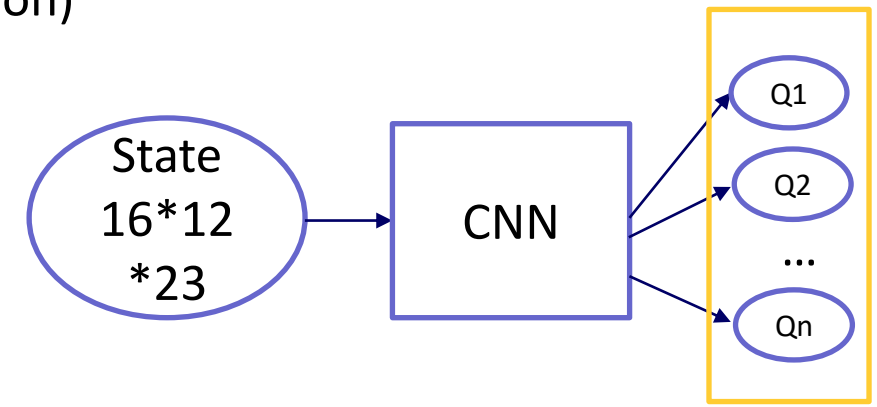
- **Position** (1-6) of train units on the track: Boolean
- Required **internal cleaning time** of train units: Float (x/60)
- Is a train unit **under internal cleaning**: Boolean
- **Length** of train units: Float (x/500)
- **Time to arrival** of train units : Float (x/720)
- Is it the **arrival** time of a train unit: Boolean
- Next 3 **departure time** of the same material type: Float (x/720)
- Is it the **departure** time of the same material type: Boolean

Action Design (Output dimensions of NN)

- 52 track to track movements
  - 8 parking to gate
  - 8 gate to parking
  - (4 parking + 1 relocation) to 2 cleaning
  - 2 cleaning to (4 parking+1 relocation)
  - 8 parking to 1 relocation
  - 1 relocation to 8 parking
- 1 wait



Number of actions



Trigger Design (Generate Learning Events to NN)

- Arrival trigger: train and time
- Departure trigger: material and time
- End of Activity trigger: train and time
- Time trigger: every one hour

Reward Design (Generate Feedback to NN)

- Negative rewards:
  - Relocation: -0.3
  - Move to cleaning track while no cleaning required: -0.5
- Positive rewards:
  - Right departure: +2.5
  - Arrival on time: +0.2
  - Wait for service to end:+duration/60
  - End service: +duration/60
  - Find a solution: +5
- Violations: cost a life
  - Lost 3 continuous lives or no available actions: end the episode

Violations

- Choose start **track** that **is empty**
- Choosing to wait in time for arrival or departure
- **Parking** a train **on** track **relocation** track or **gate** track
- Choosing **wrong time** for departure
- Choosing **wrong type** for departure
- Choosing **not clean train** for departure
- **Moving** train while **in service**
- Track **length violation**
- **Missing** a **departure** or **arrival** while doing other movements

From Q network to Value network

$$Q^{\pi}(s,a) = \mathbb{E}\left[R_t \middle| s_t = s, a_t = a, \pi\right], \text{ and}$$
$$V^{\pi}(s) = \mathbb{E}_{a\sim\pi(s)}\left[Q^{\pi}(s,a)\right].$$

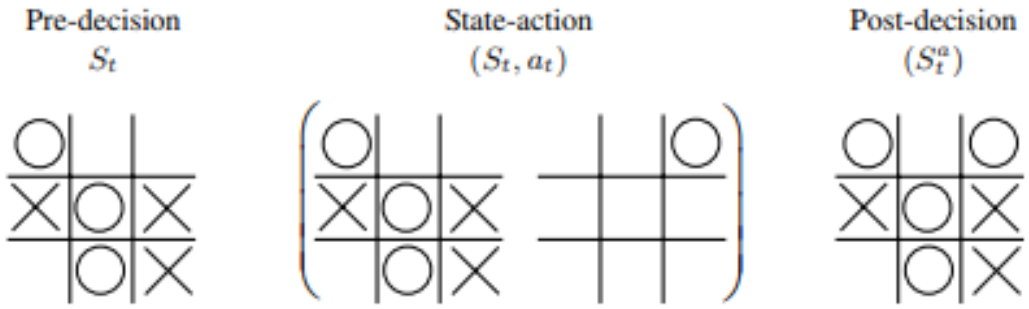
- TUSP agent has a deterministic policy

$$a = \arg\max_{a' \in \mathcal{A}} Q^*(s,a')$$

It follows

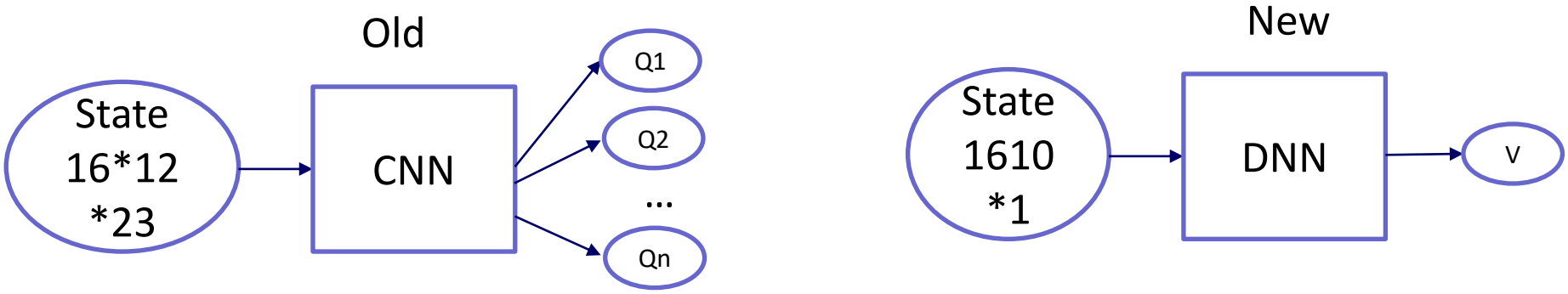
$$V^*(s) = \max_a Q^*(s,a)$$

Post-decision state variable



Value Iteration with Post-decision State (VIPS)

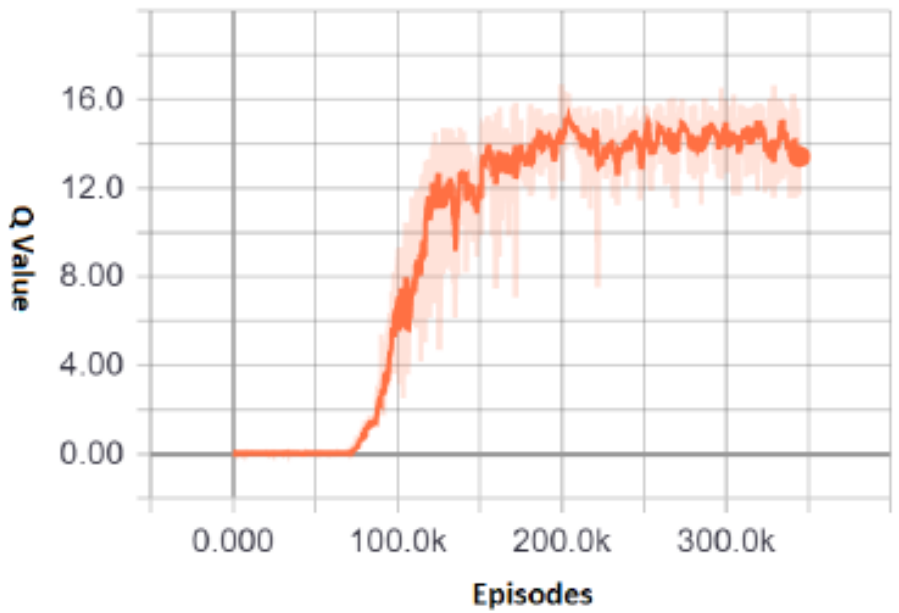
- Reduce the output dimension from 53 to 1
- Instead of estimating Q values of 53 actions (52 movement + 1 wait) at one time, estimate only the V value of the given state.



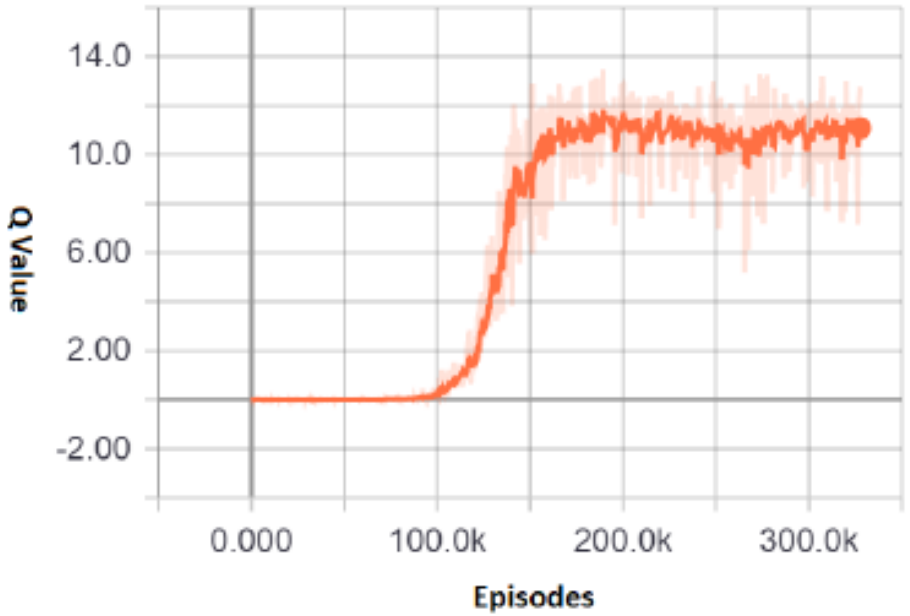
Experiment

- Instance Generation:
  - 5,000 problem instances are generated for 4, 5, 6 and 7 trains each
  - From these 20,000 problem instances, 1,000 are randomly withdraw as the test instance while the rest are used for training the DRL agents.
  - The shunting yard studied in this work is ’de Kleine Binckhorst’
- Neural network architecture
  - 2 dense hidden layer of 256 and 128 nodes, separately, with ReLu activation function
  - Output of DQN: 53 dimensional vector; output of VIPS: 1 dimensional vector

Performance: Convergence



Q values of VIPS learned on all actions



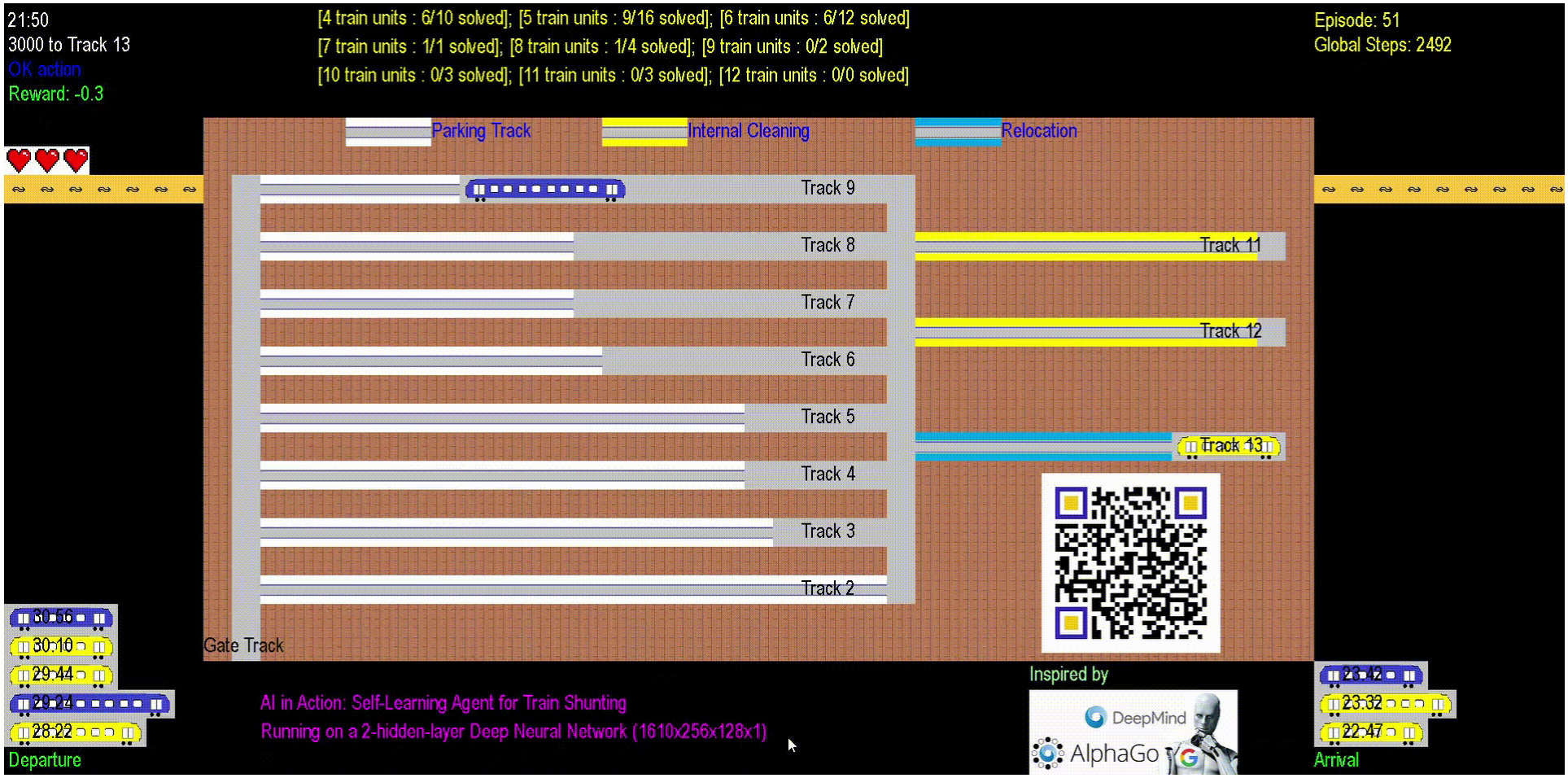
Q values of DQN learned on all actions

Performance: Problem Solving Capability

	Percentage of solved instances
VIPS, filtered actions	0.601±0.030
VIPS, all actions	0.905±0.037
DQN, filtered actions	0±0
DQN, all actions	0.782±0.016

Average percentage of solved instances and standard deviations of different models on solving 5 sets of 200 test instances.

Visualization of a TUSP reinforcement agent



Q&A

Further interets/questions: wan-jui.lee@ns.nl