



AI Approach to Bike Rebalancing using Reinforcement Learning

Ian Xiao, Alex Shannon,
Prince Abunku, Brenton Arnaboldi

What is the problem?

Citi Bike has a rebalancing problem due to the growing size of operation and travel patterns ...

- 12,000 bikes in NYC, 752 active stations, average 30,111 rides per day
- Asymmetric shift of bikes from popular originations to common destinations
- Pressing problem to scale rebalancing workforce as the program expands



Image Sources: see ppt notes.

Can we develop an intelligent solution to show what is the best strategy of moving bikes? Yes, Reinforcement Learning can help.

What is Reinforcement Learning (RL)?

Training a Reinforcement Learning model is similar to how humans and animals learn “good behaviors”.

Agent (e.g. person) selects an action based on the environment, and receives a reward or punishment based on their action.

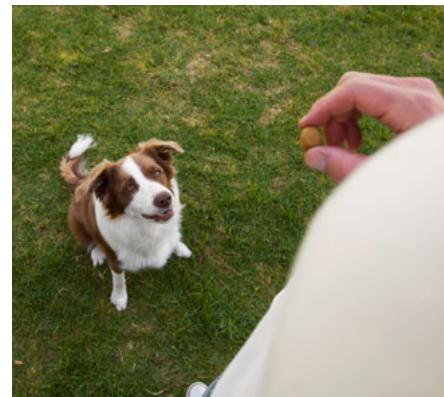


Image Sources:

<https://www.independent.co.uk/life-style/health-and-families/how-to-decide-if-you-want-kids-or-not-a7752011.html>
<http://www.vetstreet.com/our-pet-experts/teach-your-dog-to-sit>

Why use Reinforcement Learning?

Imagine using the traditional supervised and unsupervised learning techniques to achieve the following:

- Learning without explicit human instructions
- Continuous improvement over time
- Capture and adjust to complex and changing system dynamics
- Learn without deliberate feature engineering

The traditional feature engineering, predictive modelling, and cross validation approach are not designed to solve these problems.

How is RL being applied?

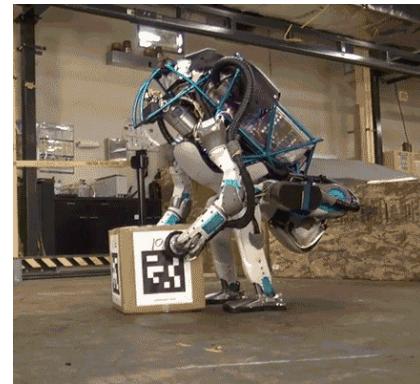


Image Sources:

<https://www.dezeen.com/2018/01/10/self-driving-cars-exacerbate-congestion-says-paul-priestman/>

<https://deepmind.com/research/alphago/>

<https://postandparcel.info/71426/news/googles-boston-dynamics-releases-video-of-parcel-handling-robot/>

What is our set up?

We want to design an intelligent solution that can do the following **without explicit human instructions**:

- Maintain bike stocks within a certain range (0 to 50)
- Understand what the bike stock requirements are
- Decide how many bikes to move to meet the requirements
- Determine when to move bikes

By trial and error, the solution should meet the following **objectives**:

- Higher success ratio: keep bike stock within a certain range (0 - 50 bikes by hour 23)
- Lower cost: move the least amount of bikes possible

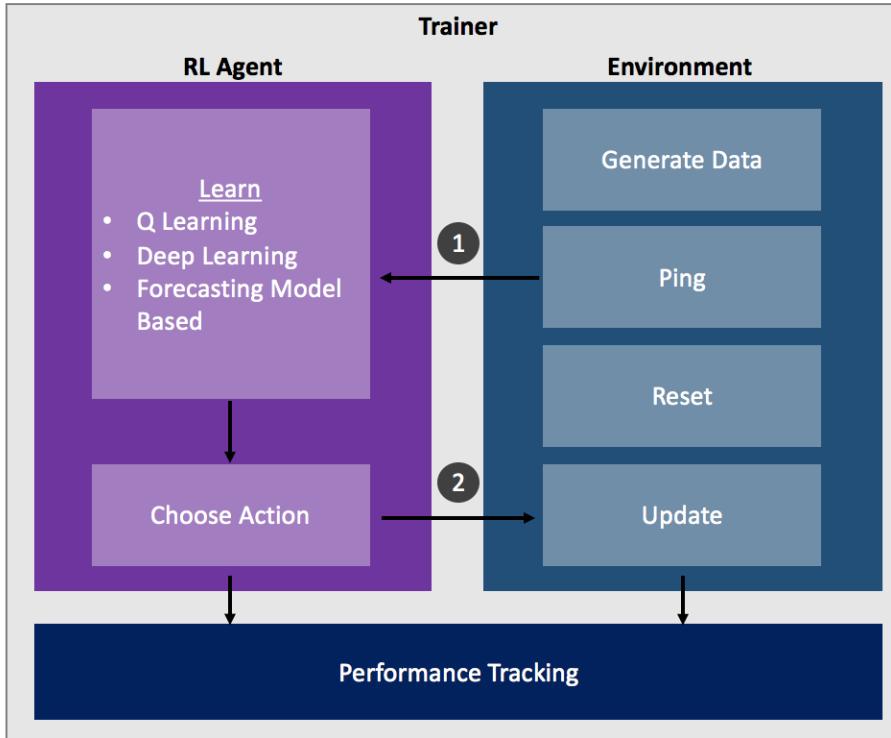
Environment: bike station, current number of bikes at station, and the current hour

Action Space: move bikes, or do nothing

"Reward" Per Hour: -30 if bike stock is below 0 or above 50 ,

-0.5*(number of bikes moved)

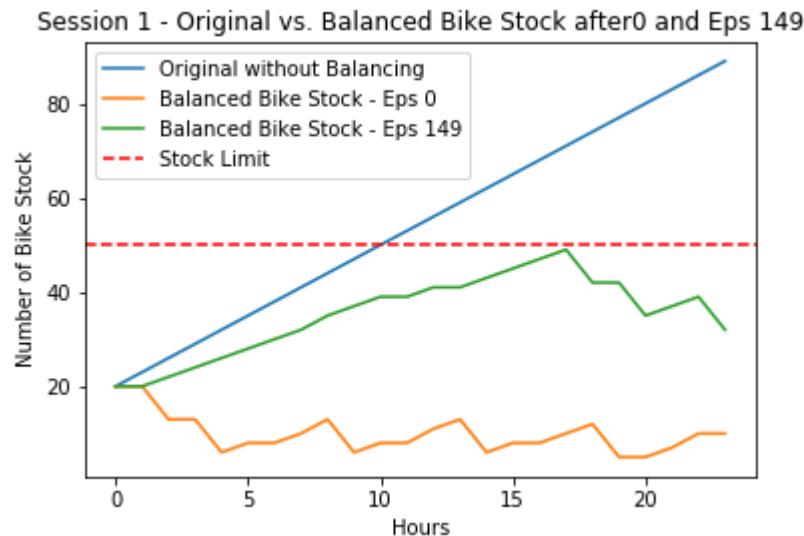
What is our solution?



- 1 Environment Feedback**
 - State (# of bike stock)
 - Reward / Penalty
 - Future State (# of new bike stock after agent action)
 - Terminate Signal
- 2 Agent Decision**
 - Action (# of bikes to move)

What are our results?

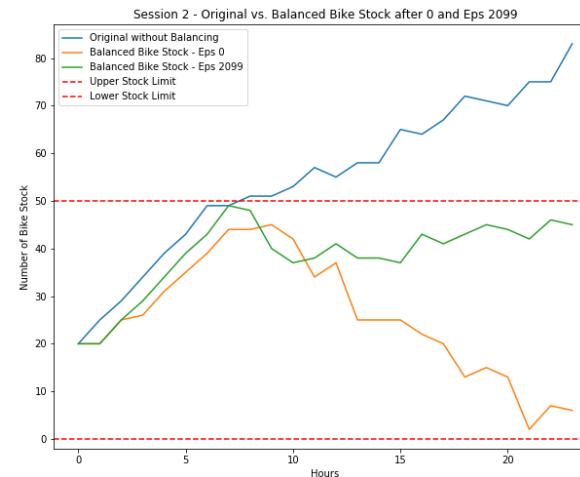
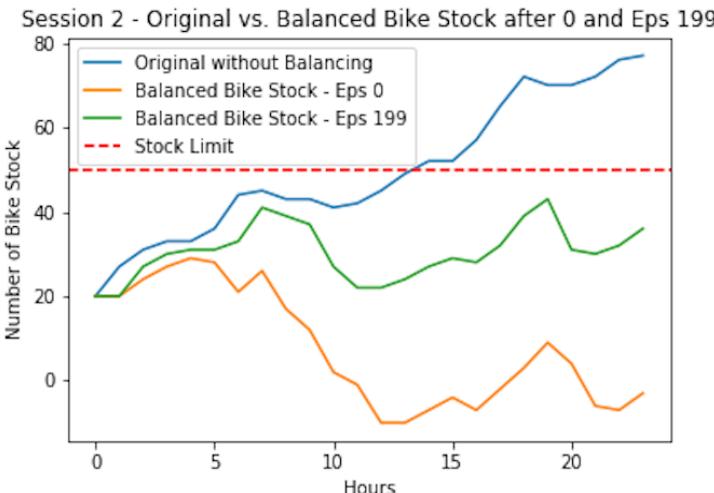
- Can the RL agent recognize a upper bike stock bound? Yes.
- Can the RL agent decide when to move and not to move bikes to save cost? Yes.
- Does the RL agent find a better strategy by learning over time? Yes.



What are our results?

Let's make it more realistic and complicated ...

- Can the RL agent relearn when there are unpredictability? Yes.
- Can the RL agent recognize the new lower bound and respond accordingly? Yes.



What are our results?

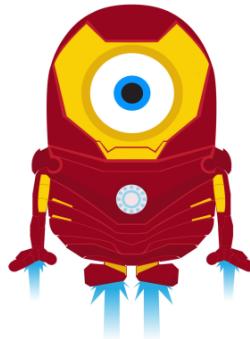
Can we understand how the RL agent makes decisions?

Actions [-10, -3, -1, 0]

s	a	w ₀	w ₁	w ₂	w ₃	w ₄
-4	0	-0.014908	0	-0.019983	0	0
-2	0	0	-0.020978	0	0	0
-1	0	0	-0.00199	0.084414	0	0
0	0	0	-0.001	0.00099	0.00099	0
1	0	0	-0.001	-0.00199	0	0
2	0	0	0	-0.021965	0	0
3	0	0	0	-0.00099	-0.00099	0
4	0	0	-0.003	-0.002948	0.01245208	0
5	0	-0.019	-0.007249	0.016896	0.01245208	0
6	0	-0.01	-0.002948	0.006271	-0.0555982	0
7	0	-0.01	-0.011852	0.006271	-0.0555982	0
8	0	-0.019	-0.002948	0.016896	0.01245208	0
9	0	-0.019	-0.03783	0.229499	0.0589517	0
10	0	0.233436	-0.0020925	0.229499	0.0589517	0
11	0	0.233436	-0.0020925	0.060282	0.0589517	0
12	-0.420768	0.0944121	1.077342	0.404824	0	0
13	0	0.660929	0.4282695	0.6787328	1.991302	0
14	0	0.660929	0.4282695	0.6787328	1.991302	0
15	0	-0.081541	0.0623028	1.742253	2.200948	0
16	0	-0.081541	0.0623028	1.742253	2.200948	0
17	0	-0.117209	0.1169319	2.241329	3.4039112	0
18	-0.2150887	0.0545099	3.7373428	2.9557177	0	0
19	0	0.233436	0.233436	0.233436	0.233436	0
20	0	0.771729	3.349528	3.63809	3.8869282	0
21	0.2337268	2.2301311	3.8459727	3.4060109	0	0
22	0	0.2337268	2.2301311	3.8459727	3.4060109	0
23	1.917994	3.7886469	4.0443076	4.3777831	0	0
24	0	2.7270212	4.787238	5.1999343	5.4734724	0
25	0	3.2679798	5.3266972	6.1976777	5.9599886	0
26	0	3.2480642	5.3266972	6.2001164	6.2001164	0
27	0	3.2480642	7.411574	8.1159971	8.292817	0
28	0	3.9931974	5.6296185	6.1602487	6.7175827	0
29	0	4.3612871	5.9491974	6.6997426	7.076141	0
30	0	4.3612871	6.2001164	7.095132	7.4639966	0
31	0	4.3612871	6.2001164	7.095132	7.4639966	0
32	0	4.821381	6.3737886	6.8201164	7.3464622	0
33	0	5.9641344	7.1195641	5.9016278	5.8911519	0
34	0	5.7220624	7.8150308	6.452009	6.0400729	0
35	0	4.8486339	6.1476746	7.3526343	6.3506098	0
36	0	5.7211819	7.1115616	7.3526343	6.0456118	0
37	0	5.8454847	6.0660138	5.8466178	5.8381591	0
38	0	7.022391	6.1303269	6.103348	5.6848608	0
39	0	5.370472	5.0593261	6.3725795	4.6408963	0
40	0	5.9309883	3.3287993	4.2585474	3.52284	0
41	0	5.2907647	3.1803556	3.6720038	1.0810643	0
42	0	5.4854064	2.7729352	2.5440557	3.0909758	0
43	0	5.2907647	2.7729352	2.5440557	3.0909758	0
44	0	5.2907647	2.7729352	2.5440557	3.0909758	0
45	0	5.2907647	2.7729352	2.5440557	3.0909758	0
46	0	5.2907647	2.7729352	2.5440557	3.0909758	0
47	0	5.2907647	2.7729352	2.5440557	3.0909758	0
48	0	5.2907647	2.7729352	2.5440557	3.0909758	0
49	0	5.2907647	2.7729352	2.5440557	3.0909758	0
50	0	5.2907647	2.7729352	2.5440557	3.0909758	0
51	0	5.2907647	2.7729352	2.5440557	3.0909758	0
52	0	5.2907647	2.7729352	2.5440557	3.0909758	0
53	0	5.2907647	2.7729352	2.5440557	3.0909758	0
54	0	5.2907647	2.7729352	2.5440557	3.0909758	0
55	0	5.2907647	2.7729352	2.5440557	3.0909758	0
56	0	5.2907647	2.7729352	2.5440557	3.0909758	0
57	0	5.2907647	2.7729352	2.5440557	3.0909758	0
58	0	5.2907647	2.7729352	2.5440557	3.0909758	0
59	0	5.2907647	2.7729352	2.5440557	3.0909758	0
60	0	5.2907647	2.7729352	2.5440557	3.0909758	0
61	0	5.2907647	2.7729352	2.5440557	3.0909758	0
62	0	5.2907647	2.7729352	2.5440557	3.0909758	0
63	0	5.2907647	2.7729352	2.5440557	3.0909758	0
64	0	5.2907647	2.7729352	2.5440557	3.0909758	0
65	0	5.2907647	2.7729352	2.5440557	3.0909758	0
66	0	5.2907647	2.7729352	2.5440557	3.0909758	0
67	0	5.2907647	2.7729352	2.5440557	3.0909758	0
68	0	5.2907647	2.7729352	2.5440557	3.0909758	0
69	0	5.2907647	2.7729352	2.5440557	3.0909758	0
70	0	5.2907647	2.7729352	2.5440557	3.0909758	0
71	0	5.2907647	2.7729352	2.5440557	3.0909758	0
72	0	5.2907647	2.7729352	2.5440557	3.0909758	0
73	0	5.2907647	2.7729352	2.5440557	3.0909758	0
74	0	5.2907647	2.7729352	2.5440557	3.0909758	0
75	0	5.2907647	2.7729352	2.5440557	3.0909758	0
76	0	5.2907647	2.7729352	2.5440557	3.0909758	0
77	0	5.2907647	2.7729352	2.5440557	3.0909758	0
78	0	5.2907647	2.7729352	2.5440557	3.0909758	0
79	0	5.2907647	2.7729352	2.5440557	3.0909758	0
80	0	5.2907647	2.7729352	2.5440557	3.0909758	0
81	0	5.2907647	2.7729352	2.5440557	3.0909758	0
82	0	5.2907647	2.7729352	2.5440557	3.0909758	0
83	0	5.2907647	2.7729352	2.5440557	3.0909758	0
84	0	5.2907647	2.7729352	2.5440557	3.0909758	0
85	0	5.2907647	2.7729352	2.5440557	3.0909758	0
86	0	5.2907647	2.7729352	2.5440557	3.0909758	0
87	0	5.2907647	2.7729352	2.5440557	3.0909758	0
88	0	5.2907647	2.7729352	2.5440557	3.0909758	0
89	0	5.2907647	2.7729352	2.5440557	3.0909758	0
90	0	5.2907647	2.7729352	2.5440557	3.0909758	0
91	0	5.2907647	2.7729352	2.5440557	3.0909758	0
92	0	5.2907647	2.7729352	2.5440557	3.0909758	0
93	0	5.2907647	2.7729352	2.5440557	3.0909758	0
94	0	5.2907647	2.7729352	2.5440557	3.0909758	0
95	0	5.2907647	2.7729352	2.5440557	3.0909758	0
96	0	5.2907647	2.7729352	2.5440557	3.0909758	0
97	0	5.2907647	2.7729352	2.5440557	3.0909758	0
98	0	5.2907647	2.7729352	2.5440557	3.0909758	0
99	0	5.2907647	2.7729352	2.5440557	3.0909758	0
100	0	5.2907647	2.7729352	2.5440557	3.0909758	0
101	0	5.2907647	2.7729352	2.5440557	3.0909758	0
102	0	5.2907647	2.7729352	2.5440557	3.0909758	0
103	0	5.2907647	2.7729352	2.5440557	3.0909758	0
104	0	5.2907647	2.7729352	2.5440557	3.0909758	0
105	0	5.2907647	2.7729352	2.5440557	3.0909758	0
106	0	5.2907647	2.7729352	2.5440557	3.0909758	0
107	0	5.2907647	2.7729352	2.5440557	3.0909758	0
108	0	5.2907647	2.7729352	2.5440557	3.0909758	0
109	0	5.2907647	2.7729352	2.5440557	3.0909758	0
110	0	5.2907647	2.7729352	2.5440557	3.0909758	0
111	0	5.2907647	2.7729352	2.5440557	3.0909758	0
112	0	5.2907647	2.7729352	2.5440557	3.0909758	0
113	0	5.2907647	2.7729352	2.5440557	3.0909758	0
114	0	5.2907647	2.7729352	2.5440557	3.0909758	0
115	0	5.2907647	2.7729352	2.5440557	3.0909758	0
116	0	5.2907647	2.7729352	2.5440557	3.0909758	0
117	0	5.2907647	2.7729352	2.5440557	3.0909758	0
118	0	5.2907647	2.7729352	2.5440557	3.0909758	0
119	0	5.2907647	2.7729352	2.5440557	3.0909758	0
120	0	5.2907647	2.7729352	2.5440557	3.0909758	0
121	0	5.2907647	2.7729352	2.5440557	3.0909758	0
122	0	5.2907647	2.7729352	2.5440557	3.0909758	0
123	0	5.2907647	2.7729352	2.5440557	3.0909758	0
124	0	5.2907647	2.7729352	2.5440557	3.0909758	0
125	0	5.2907647	2.7729352	2.5440557	3.0909758	0
126	0	5.2907647	2.7729352	2.5440557	3.0909758	0
127	0	5.2907647	2.7729352	2.5440557	3.0909758	0
128	0	5.2907647	2.7729352	2.5440557	3.0909758	0
129	0	5.2907647	2.7729352	2.5440557	3.0909758	0
130	0	5.2907647	2.7729352	2.5440557	3.0909758	0
131	0	5.2907647	2.7729352	2.5440557	3.0909758	0
132	0	5.2907647	2.7729352	2.5440557	3.0909758	0
133	0	5.2907647	2.7729352	2.5440557	3.0909758	0
134	0	5.2907647	2.7729352	2.5440557	3.0909758	0
135	0	5.2907647	2.7729352	2.5440557	3.0909758	0
136	0	5.2907647	2.7729352	2.5440557	3.0909758	0
137	0	5.2907647	2.7729352	2.5440557	3.0909758	0
138	0	5.2907647	2.7729352	2.5440557	3.0909758	0
139	0	5.2907647	2.7729352	2.5440557	3.0909758	0
140	0	5.2907647	2.7729352	2.5440557	3.0909758	0
141	0	5.2907647	2.7729352	2.5440557	3.0909758	0
142	0	5.2907647	2.7729352	2.5440557	3.0909758	0
143	0	5.2907647	2.7729352	2.5440557	3.0909758	0
144	0	5.2907647	2.7729352	2.5440557	3.0909758	0
145	0	5.2907647	2.7729352	2.5440557	3.0909758	0
146	0	5.2907647	2.7729352	2.5440557	3.0909758	0
147	0	5.2907647	2.7729352	2.5440557	3.0909758	0
148	0	5.2907647	2.7729352	2.5440557	3.0909758	0
149	0	5.2907647	2.7729352	2.5440557	3.0909758	0
150						

Can we improve the learning performance?

Q-Learning
(what you saw)



- Markov Decision Process using a matrix to keep track of learning
- The most simple form of learning
- Does not scale

Q-Learning +
Forecasting



- Forecast bike stock in the next hour using a Random Forests Model
- Incorporate the forecast into decision and learning

Deep Q Learning



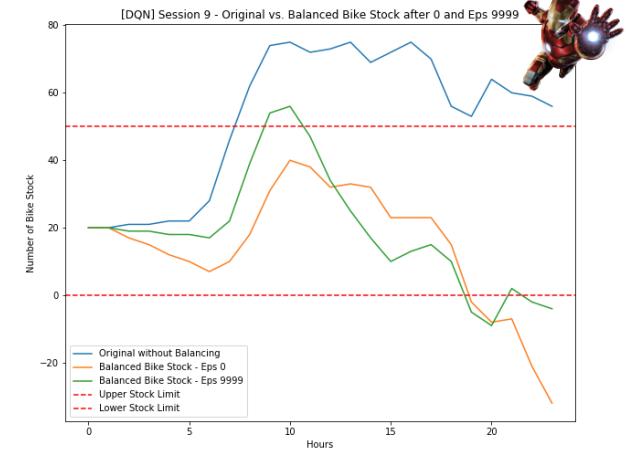
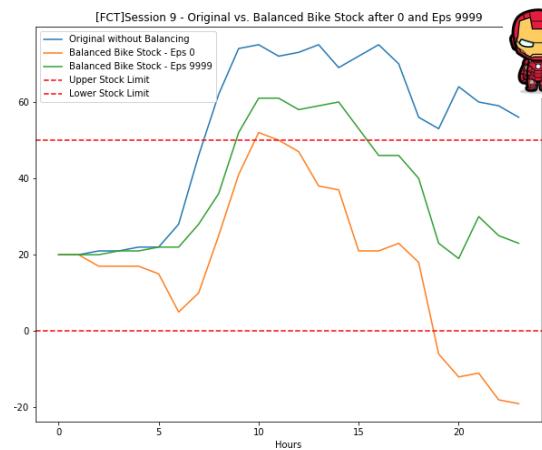
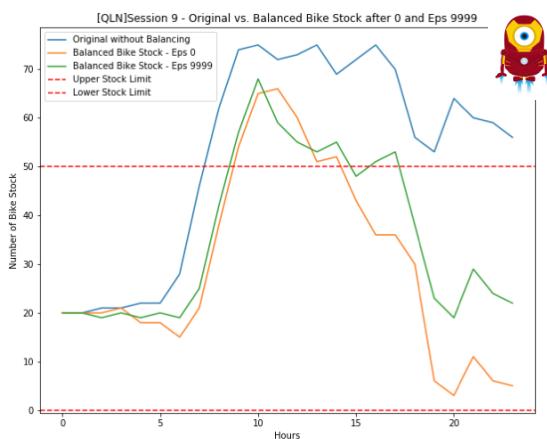
- Use Recurrent Neural Network to remember the learning
- Scale for large system parameters
- Minimizes computation and memory use

What are our results?

Let's make it even more realistic by using actual Citi Bike data and benchmark across all three learning methods.

Goal: keep bike stock balanced between [0, 50] while moving the fewest bikes possible.

Bike Stock at 17th and Broadway (North Edge of Union Square) on 9/1/2017

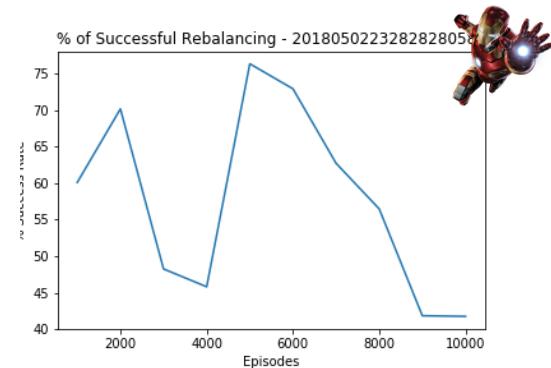
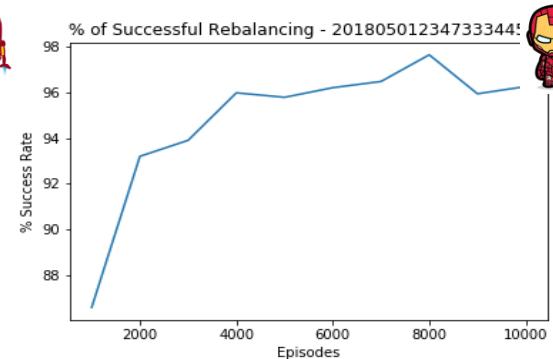
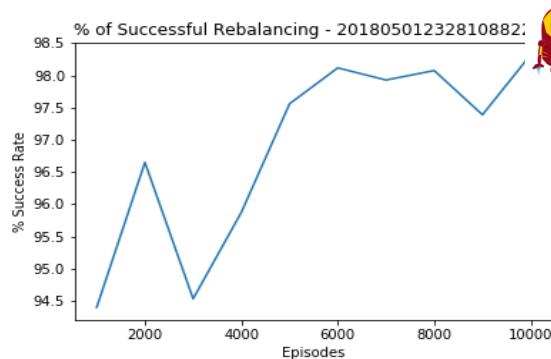


What are our results?

How does each learning method improve over time after more interaction with the environment?

- Some methods shown improvement in success rate (keeping bikes between 0 and 50 at hour 23)
- Some variation in performance consistency

Evaluation Metric: % of episodes where final bike stock is in range [0, 50]



Next Steps

- Augment the solution to rebalance bicycles across multiple stations
- Expand the action space from just removing bikes to adding bikes
- Enable parallel computing to shorten training time
- Develop better tuned forecasting and Deep Learning models
- Apply trained model to different (future) dates