Freie Universität Bozen
Libera Università di Bolzano
Università Liedia de Bulsan

Federated Reinforcement Learning on the Edge

Hannes Wiedenhofer

Supervisor: Dr. Roberto Confalonieri

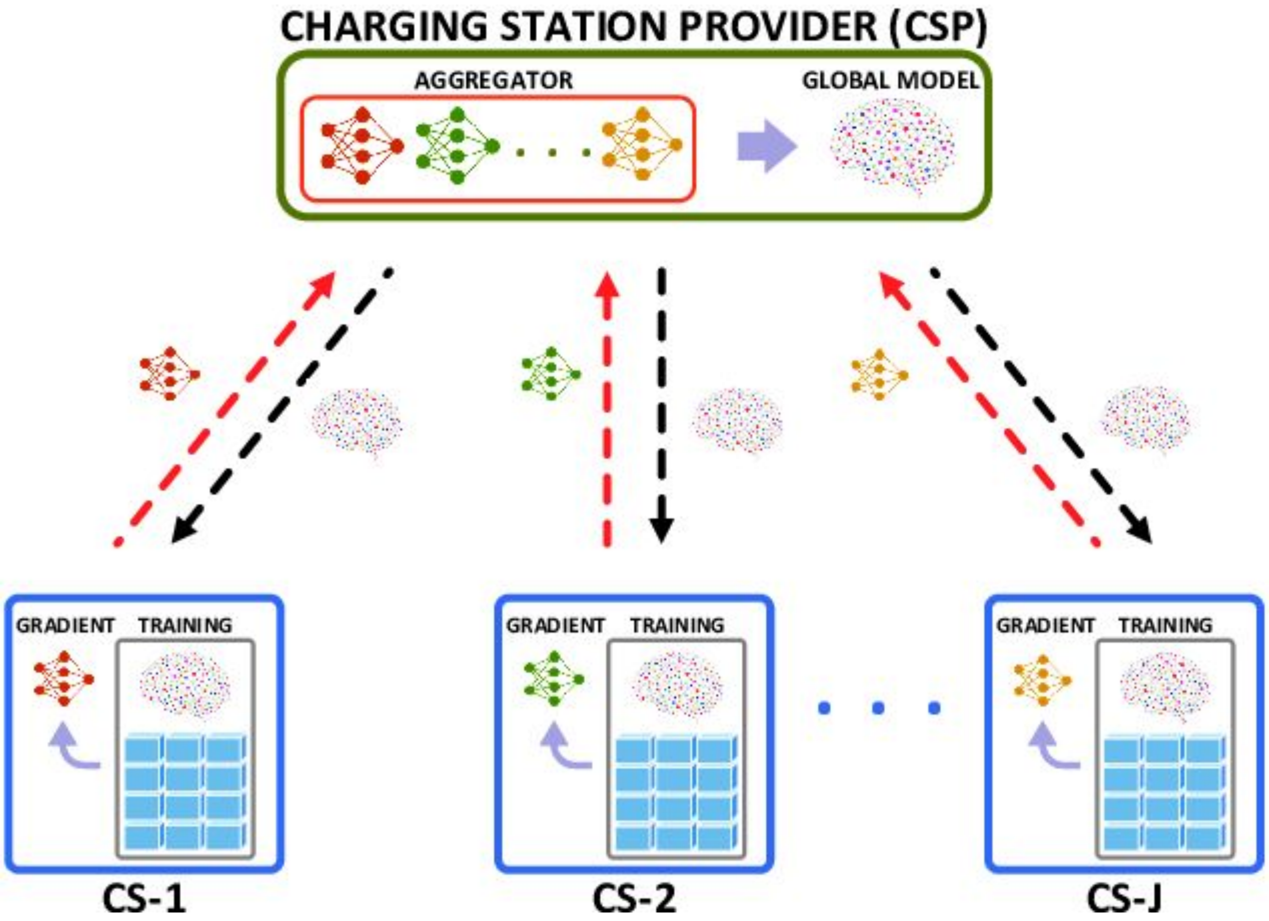
Co-Supervisor: Prof. Antonio Liotta

Outline

- Introduction
- Problem Statement
- Background Information
- Problem Solution
- Evaluation
- Conclusion and Further Studies

Motivation

- Energy demand prediction for ev charging stations
- Local computational resources limited



Introduction

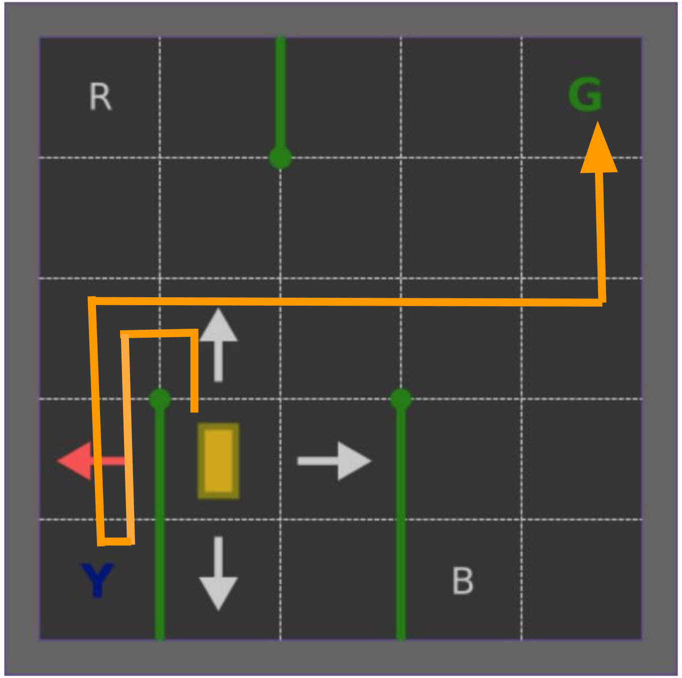
- Emergence of IoT and privacy consciousness
- Restrictions: computational, memory, network
- Objective: create solid machine learning model for multiple agents
- How? Conceptual federated reinforcement learning framework
- Approach:
  - analyze existing federated learning frameworks
  - implement our own conceptual federated learning framework with different architectures

Problem Statement

- Develop a conceptual federated learning framework that works on the edge
- Edge devices: memory restrictions, computational restrictions
- Q-tables must be kept as small as possible
- Model quality should still be decent

Background Information: Reinforcement Learning

- Agent learns by trial and error
- Components:
    - **Agent:** acts within environment, learns based on its actions (Taxi)
    - **Environment:** everything the agent interacts with, assigns rewards (Grid world)
    - **Action:** the agent's means of interacting with the environment (moving left, right, up, down, picking up passenger, dropping off passenger)
    - **State:** the situation the agent finds itself in (current location of taxi within grid world)
    - **Reward:** the feedback given from the environment to the agent (negative reward for crashing into wall, positive for successful ride)
    - **Policy:** mapping from a state to the probabilities of selecting each possible action given that state (orange path)



OpenAI taxi-V3 environment(Image by Guillaume Androz from towardsdata-science).

Background Information: Q-Learning

- Off-policy model-free reinforcement learning algorithm for discrete action and state spaces
- For any finite Markov Decision Process it finds a policy that is optimal
- Bellman equation:

$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\Big(\underbrace{\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}}_{\text{new value (temporal difference target)}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \Big)}^{\text{temporal difference}}$$

Background Information: Bellman Equation Example

- First learning episode -> old value = 0
- Episode not successful -> negative reward

$$Q^{new}(s_t, a_t) \leftarrow \underbrace{0}_{\text{old value}} + \underbrace{0.01}_{\text{learning rate}} \cdot \overbrace{\Big(\underbrace{\underbrace{-270}_{\text{reward}} + \underbrace{0.9}_{\text{discount factor}} \cdot \underbrace{180}_{\text{estimate of optimal future value}}}_{\text{new value (temporal difference target)}} - \underbrace{0}_{\text{old value}} \Big)}^{\text{temporal difference}}$$

Q_new = 159,3

Background Information: Q-Tables

- Agent chooses actions based on its q-table
- Initialized with all values 0
- Values are updated using the Bellman equation

Initialized

Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	327	0	0	0	0	0	0
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	499	0	0	0	0	0	0

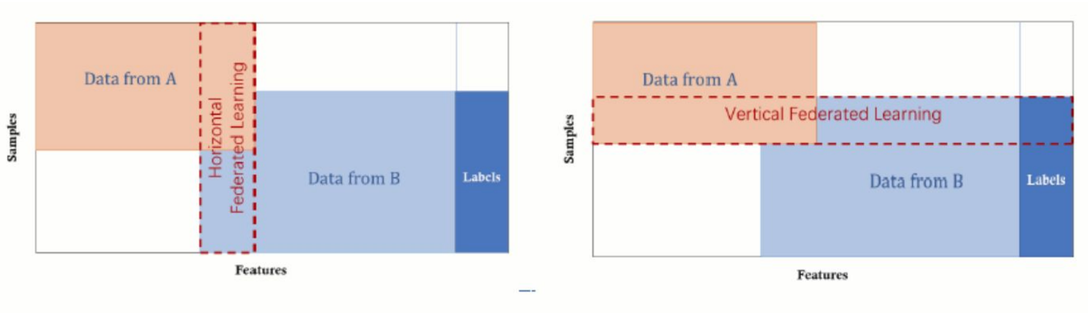
Training

Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	328	-2.30108105	-1.97092096	-2.30357004	-2.20591839	-10.3607344	-8.5583017
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	499	9.96984239	4.02706992	12.96022777	29	3.32877873	3.38230603

Example of a Q-table (Image by LearnDataSci on Wikimedia).

Background Information: Federated Learning

- A decentralized form of Machine Learning
- Aims at training machine learning models on multiple local datasets without explicitly exchanging the data
- Multiple local models are trained, parameters are exchanged to create a single global model
- Respecting (not guaranteeing) data privacy, data access rights, data security
- Two main types:



Federated Learning Frameworks: PySyft

- Open-source
- Part of the OpenMined[1] ecosystem
- Supports both vertically and horizontally partitioned data
- Security Mechanisms: MPC, HE, DP
- Supports both the PyTorch and Tensorflow libraries
- Only works in simulation mode
- Early phase of development
- No documentation
- Largest community of contributors (over 250 developers)

[1]https://www.openmined.org/

Federated Learning Frameworks: Tensorflow Federated

- Open-source
- Part of Google's Tensorflow[1] ecosystem
- Builds on Tensorflow's standard structures
- Possible aggregation functions: sum, mean, differentially private
- Only works in simulation mode
- Current version incomplete
- No decent documentation
- Dataset must be in a certain form

[1] https://www.tensorflow.org/

Why did we develop our own conceptual federated learning framework instead of using an existing framework?

- Attempt to use both PySyft and TFF
- PySyft:
  - Works with encrypted tensors via (new) Duet interface
  - Communication workflow unclear
  - Could not access the tensor's values
  - Unclear where computation happens
- TFF:
  - Followed example notebook (no documentation)
  - Model quality lower than expected
  - Due to the lack of documentation mistakes could not be found

Problem Solution: Dataset and Assumptions

- Federated reinforcement learning system
- Dataset: CitiBike[1] NYC bike sharing data
- Goal: rebalance station bike stock to stay between fixed limits
- Simulation for 24 hours
- Assumptions:
    - At each hour, each agent can move 0, 1, 3, 5, or 10 bikes from one station to another
    - The system aims at maintaining a bike stock between 0 and 50 at all times for each station
    - The initial bike stock for each station is set to 20. All stock calculations start from a stock of 20 at 0 am.

[1]https://www.citibikenyc.com/system-data

Problem Solution: Reward System and Metric

- Goal: keep stock between 0 and 50 at all times
- At each hour the agent(s) perform an action
- Based on the action, the environment assigns a reward:
    - -30 if bike stock falls outside the range [0, 50] at each hour
    - -0.5 times the number of bikes removed at each hour (avoid doing unnecessary movements because of movement costs)
    - +20 if bike stock in [0, 50] at 23 hours; else -20
- **Metric:** percentage of time where stock numbers within limits

$$success\_rate = \frac{(\#total\_h - \#h\_overstock - \#h\_understock) * 100}{\#total\_h}$$

where

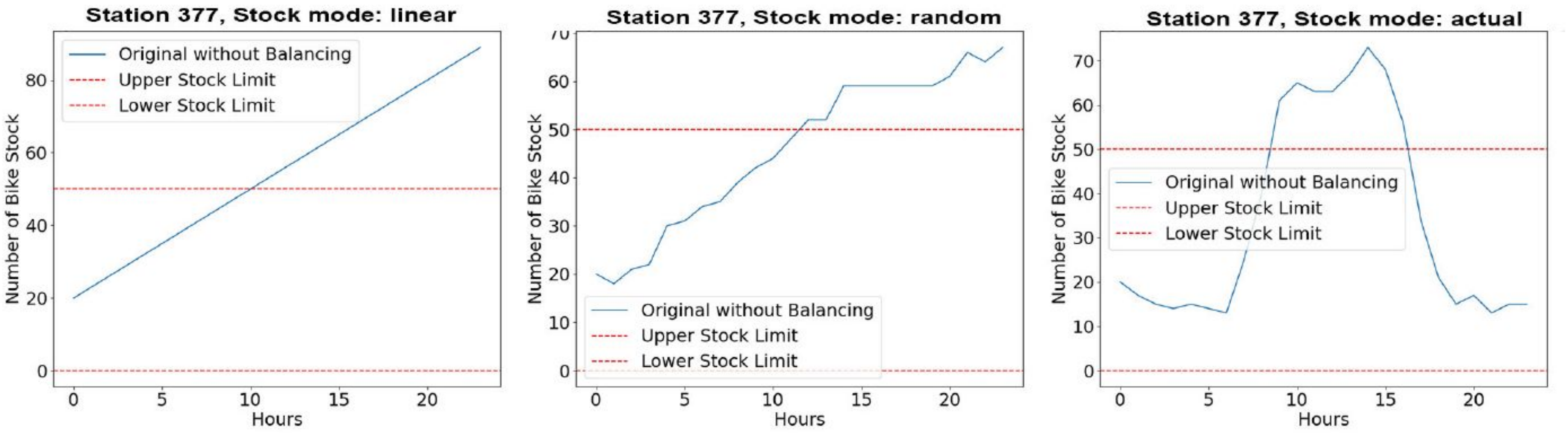
- $\#total\_h$ represents the total number of hours
- $\#h\_overstock$ is the number of hours the stock is above the upper limit
- $\#h\_understock$ is the number of hours the stock is below the lower limit

Problem Solution: Parameters 1/2

- Remove only
    - if set, the system can only remove bikes from stations, they are not assigned to other stations. They disappear.
    - Not useful in a real scenario, but helpful for determining whether the trained model works correctly
- Predictions
    - If set, enables the system to take into consideration predictions for the bike stocks.
    - Random Forests models were built for each station to create predictions.
    - The models’ ability to predict the bike stock varied from station to station

Problem Solution: Parameters 2/2

- Stock mode
    - Can be set to “linear”, “random”, or “actual”
    - Determines the original stock for the stations
    - “Linear” -> every station’s stock increases by the same amount of bikes at every hour.
    - “Random” -> similar to linear, but the number of bikes added is not constant, but random
    - “Actual” -> uses the data from the dataset to create the hourly stock numbers. Most realistic.

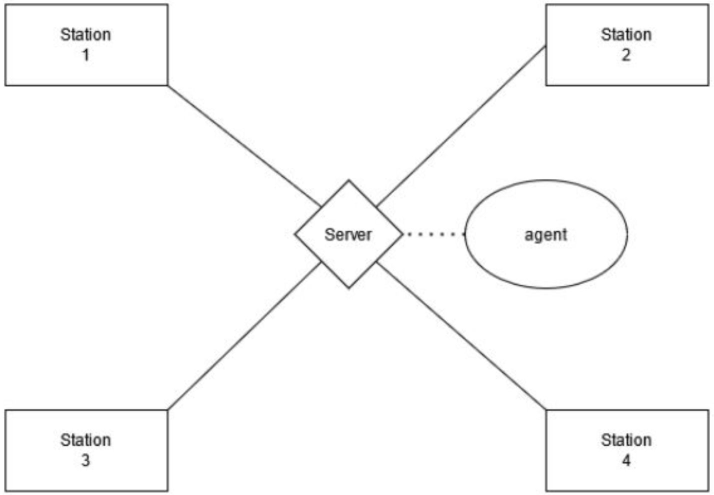


Problem Solution: Components

- Three different approaches, all have these components in common:
    - Agent: manages its q-table, chooses actions based on the q-table, learns policy as a consequence of the chosen actions
    - Environment: assigns the rewards to the agent(s), updates the stations' bike stocks, updates the simulation hour
    - Helper: reads dataset containing trip data and transforms it into the station history
    - Trainer: calls all the different elements of the system and manages the communication between them, performs the training, stores the results

Problem Solution: Centralized Approach 1/2

- Simplest approach
- Only one central agent overseeing all the stations and their bike stock
- Agent has all information about the whole system at all times



Problem Solution: Centralized Approach 2/2

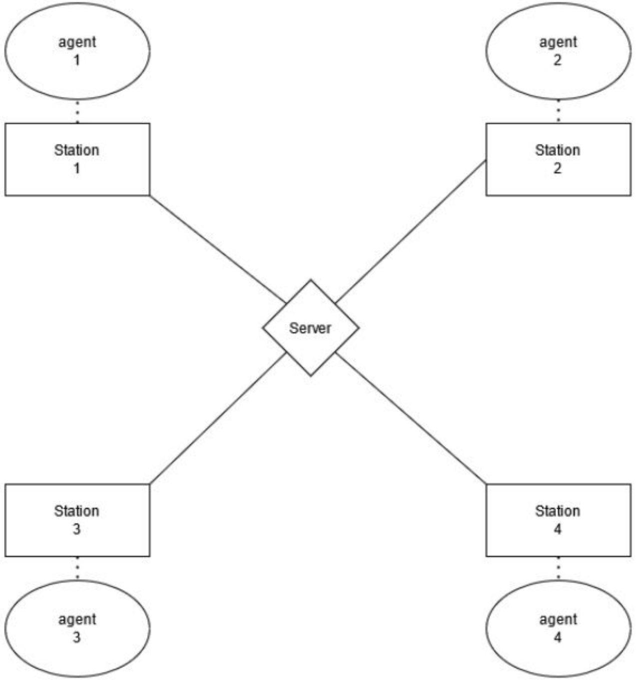
- Q-Table contains bike stock for all stations
- Large action space
- Works well for a limited amount of bike stations and limited amount of total bikes in the system
- Does not scale well

Stock	(515, 377, 1)	(377, 515, 1)	(515, 377, 3)	(377, 515, 3)	(515, 377, 5)	(377, 515, 5)	(515, 377, 10)	(377, 515, 10)	(0, 0, 0)
{"377": -3, "515": 29}	-1,14813E+16	-1,99687E+15	-1,54503E+14	-9,52336E+15	-1,03302E+16	-7,80899E+15	-1,50049E+16	-1,11053E+16	-1,0914E+16
{"377": -12, "515": 37}	-1,7556E+11	-6,19157E+15	-4,19[illegible]5E+15	-1,40728E+16	-7,12972E+15	-1,77354E+16	-4,96459E+15	-4,7117E+15	-9,3165E+14
{"377": -5, "515": 38}	-2,66119E+16	-1,95572E+15	-1,[illegible]36E+15	-1,2565E+16	-2,6139E+16	-1,50049E+16	-1,84531E+16	-1,43898E+16	-1,5155E+16
{"377": -14, "515": 45}	-2,59448E+16	-2,59448E+16	-3,13985E+15	-2,07437E+16	-5,70816E+15	-4,45627E+15	-4,96459E+15	-1,7556E+11	-9,5234E+15
{"377": -11, "515": 44}	-1,9765E+15	-2,0069E+16	-2,57308E+16	-7,80899E+15	-1,48534E+16	-1,43898E+16	-1,83365E+15	-1,89081E+15	-8,8966E+15
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Problem Solution: Distributed Approach

- Each station has its own agent
- No central entity that oversees all the bike stocks
- Server only assigns rewards
- Agents do not communicate
- Q-Table for station 377:

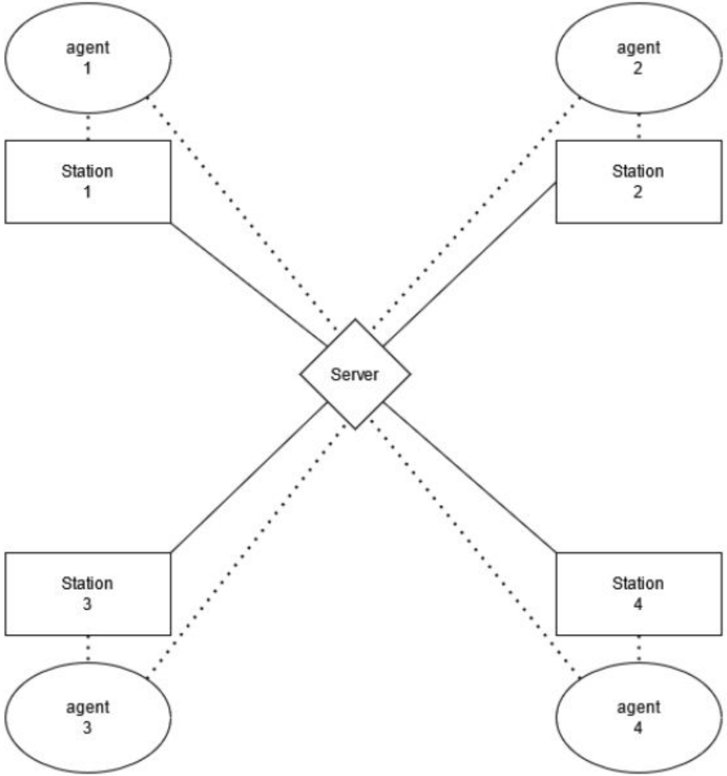
Stock	(1, 515)	(3, 515)	(5, 515)	(10, 515)	(0, 0)
20	-1,3015E+16	-3,4494E+15	-8,2757E+15	-1,3751E+16	0
27	-4,3241E+15	-1,019E+14	-2,6165E+15	-3,8628E+16	0
29	-6,1239E+16	-1,1588E+16	-742525	-6,9971E+15	0
28	-3,8628E+16	-1,2972E+16	-1,463E+12	-6,9971E+15	0
33	-5,2331E+16	-1,7042E+15	-4,5523E+16	-4,7809E+15	0
23	-5,2331E+16	-8,778E+11	-1,2252E+10	-2,926E+11	0
22	-3,8628E+16	-7351492515	-1,9314E+16	-5,2331E+15	0
41	-7,4271E+15	-2,4823E+15	-1,9314E+16	-5,2331E+15	0
61	-1,0527E+16	-1,1294E+16	-1,0527E+16	-9,7281E+13	-2,6713E+15
72	-7,5848E+15	-7,809E+15	-8,0309E+15	-8,68[illegible]+14	-1,6302E+15
75	-7,809E+15	-8,8966E+15	-1,3083E+16	-1,39[illegible]E+16	-6,1916E+15
74	-1,2031E+16	-7,1297E+15	-8,8966E+15	-7,809E+15	-1,1294E+16
8[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]



Problem Solution: Towards a Federated Approach

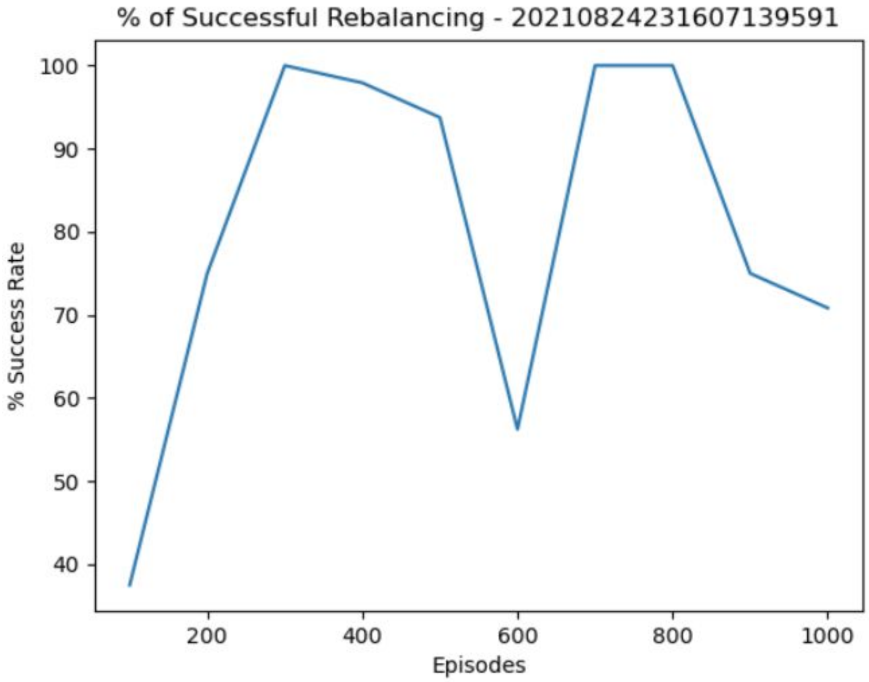
- Centralized approach does not scale well
- Distributed approach does not perform well
- Compromise

Stock	(1, 515)	(3, 515)	(5, 515)	(10, 515)	(0, 0)
{"377": 70, "515": -23}	-3,6744E+15	-4,4563E+15	-2,8685E+16	-2,3177E+15	-2,8785E+15
{"377": 76, "515": -25}	-2,8437E+15	-1,545E+14	-2,6905E+16	[illegible]	-4,9646E+15
{"377": 77, "515": -25}	-1,7556E+11	-1,3749E+15	-4,4563E+15	[illegible]	-2,3356E+16
{"377": 68, "515": -19}	-3E+16	-1,4546E+16	-3E+16	-1,8683E+16	-1,0722E+16
{"377": 69, "515": -12}	-8,4681E+15	-2,9606E+16	-7,1297E+15	-6,8987E+16	-2,9969E+16
{"377": 77, "515": -11}	-2,9551E+15	-2,9439E+16	-2,9971E+16	-2,9585E+15	-2,9055E+15
{"377": 67, "515": -1	-2,7823E+16	-2,5816E+16	-2,7909E+16	-2,6508E+16	-2,9472E+16

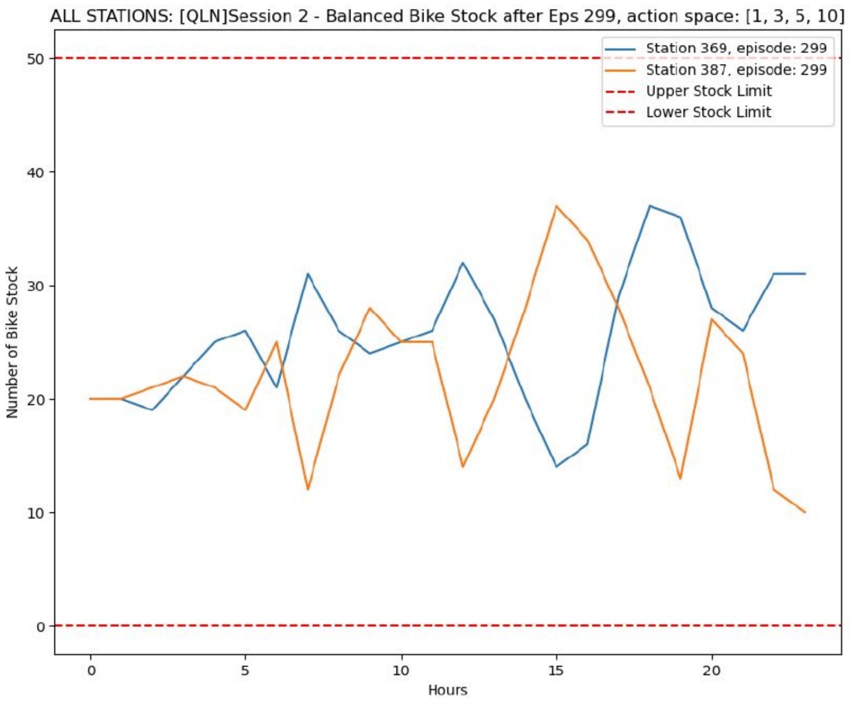


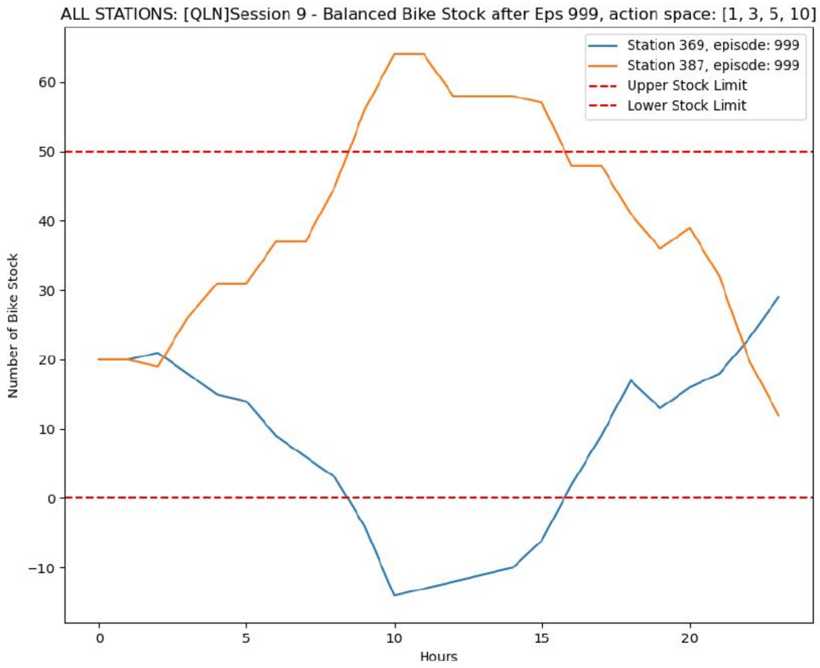
Evaluation: Methodology 1/3

- Number of episodes vs success rate
- Session 0 = 100 episodes
  Session 1 = 200 episodes
  …
  Session 9 = 1000 episodes
- Session 2 yields a better result than Session 9



Evaluation: Methodology 2/3

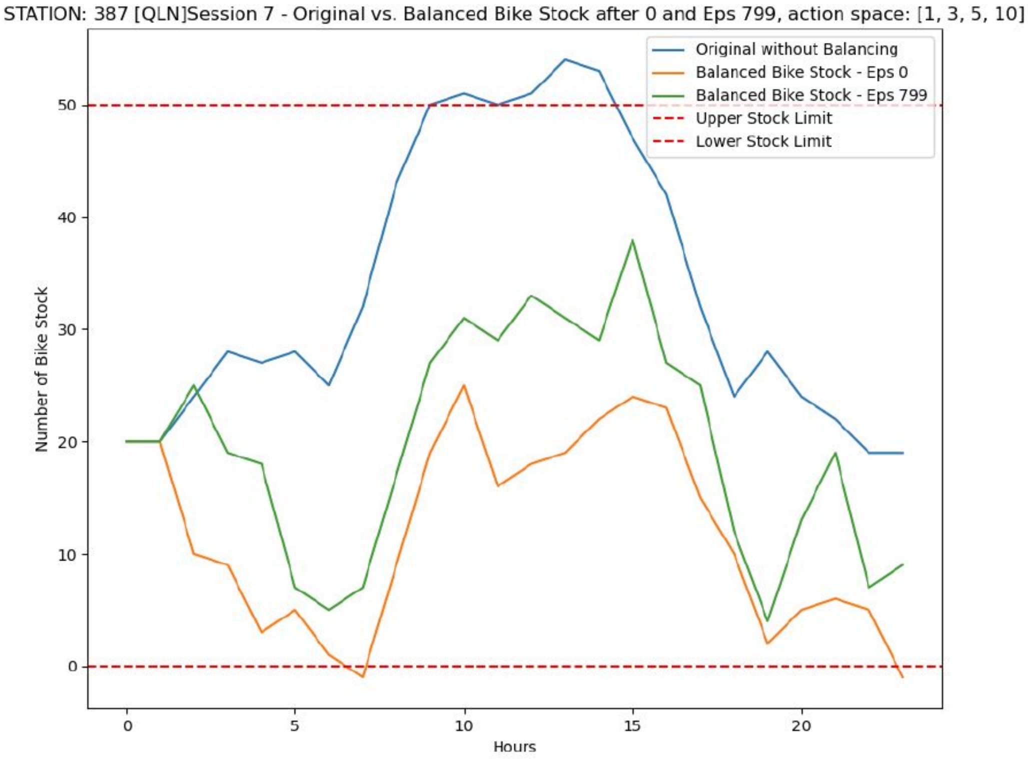




Graphs for centralized approach

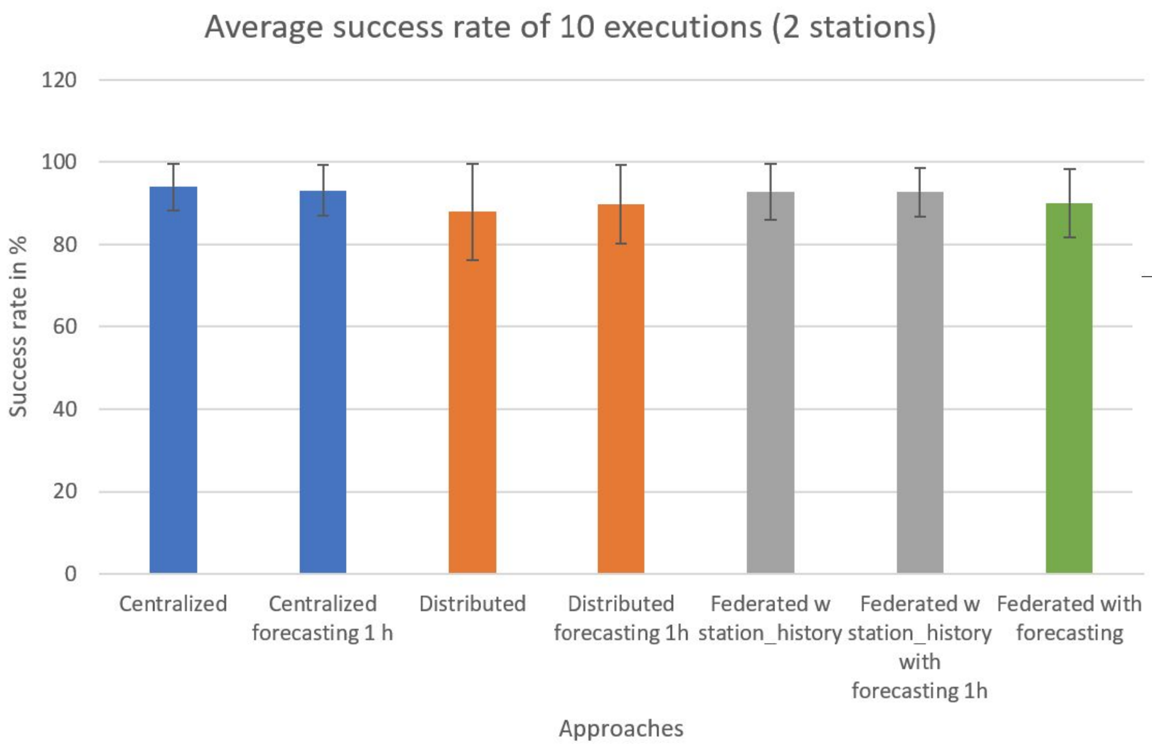
Evaluation: Methodology 3/3

- Analyzing the learning within a session
- Larger number of episodes should lead to improvement



Graph for centralized approach

Evaluation: Comparison of Results



Ranking	Success Rate	Approach
1.	93,96%	Centralized
2.	93,13%	Centralized forecasting 1h
3.	92,71%	Federated with station history
4.	92,71%	Federated with station history forecasting 1h
5.	90%	Federated with forecasting
6.	89,79%	Distributed forecasting 1h
7.	87,92%	Distributed

Conclusion and Further Studies

- We analyzed the existing federated learning frameworks
- Because of the lack of documentation and early phase of development we decided to implement our own
- We proposed a system for solving a bike rebalancing problem
- Can be adapted to function with other datasets
- Can serve as a base for building a fully federated learning system
- Analyze trade-off between communication volume and model quality

Thank you for your attention