# Reinforcement Learning:
## Algorithms comparison

Andrea Zasa

# The environments

## Frozen lake:

## Mini yahtzee:


States:

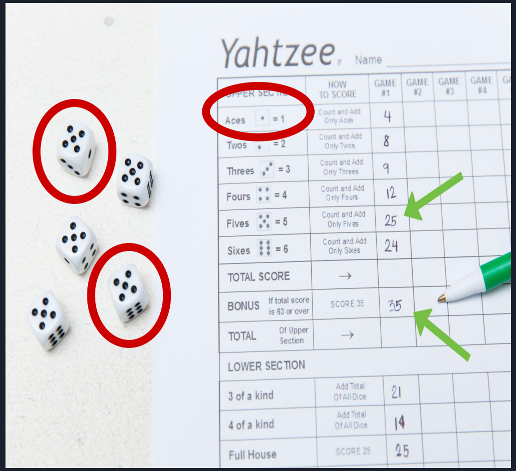
Agent position
- Unique ID
- XY coordinates
- Onehot encoding

Round, sub round, dices, score
- Unique ID
- Integers
- Onehot encoding

# The environments

Frozen lake:

Mini yahtzee:




## Actions:

Move

- Up, down, left, right

Reroll, Score

- Reroll some dices/ score category

# The environments

## Frozen lake:

## Mini yahtzee:


Transition probabilities:

- ⅓ for the selected direction
- ⅓ for the orthogonal directions

- 1 for scoring, and fair dice rules for rerolls

# The environments

## Frozen lake:


## Mini yahtzee:


Rewards

- 1 if we end up at the end, 0 otherwise

- The score we obtain

N.B there are actually some variation of this rule: punishment and reward for rolling

# Algorithms

## Tabular

TD0:
- SARSA
- EXP SARSA
- Q Learning

TDN:
- N step SARSA
- Generalized TDN

## Function approximation

TD0:
- DQN
- Double DQN

Policy Gradient:
- Reinforce with baseline

# TD0

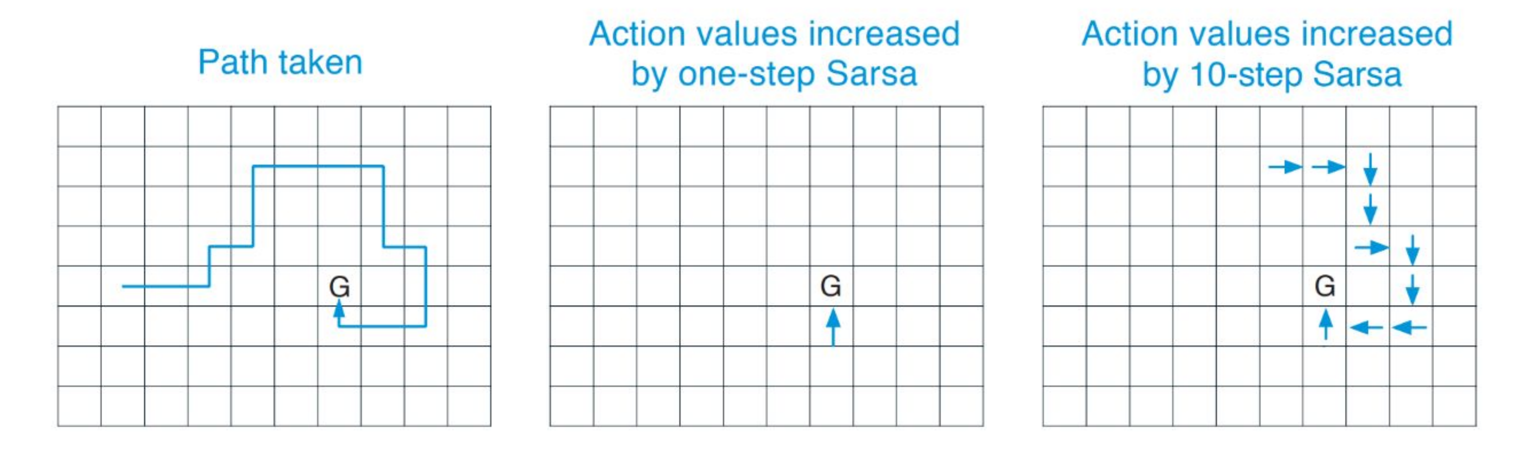
**Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$**

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize $S$
  Loop for each step of episode:
    Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
    Take action $A$, observe $R$, $S'$
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until $S$ is terminal

- All TD0 algorithms are pretty similar
- Can be done both tabular and with function approximation (N.B deadly triad)
- Multiple variations (memory/model, double, afterstates)

# TDN




- Multiple step propagation
- Difference for off policy is significant
- Higher variance, lower bias (limit is MC)

Reinforcement learning, Richard S. Sutton pag. 157

# Policy gradient

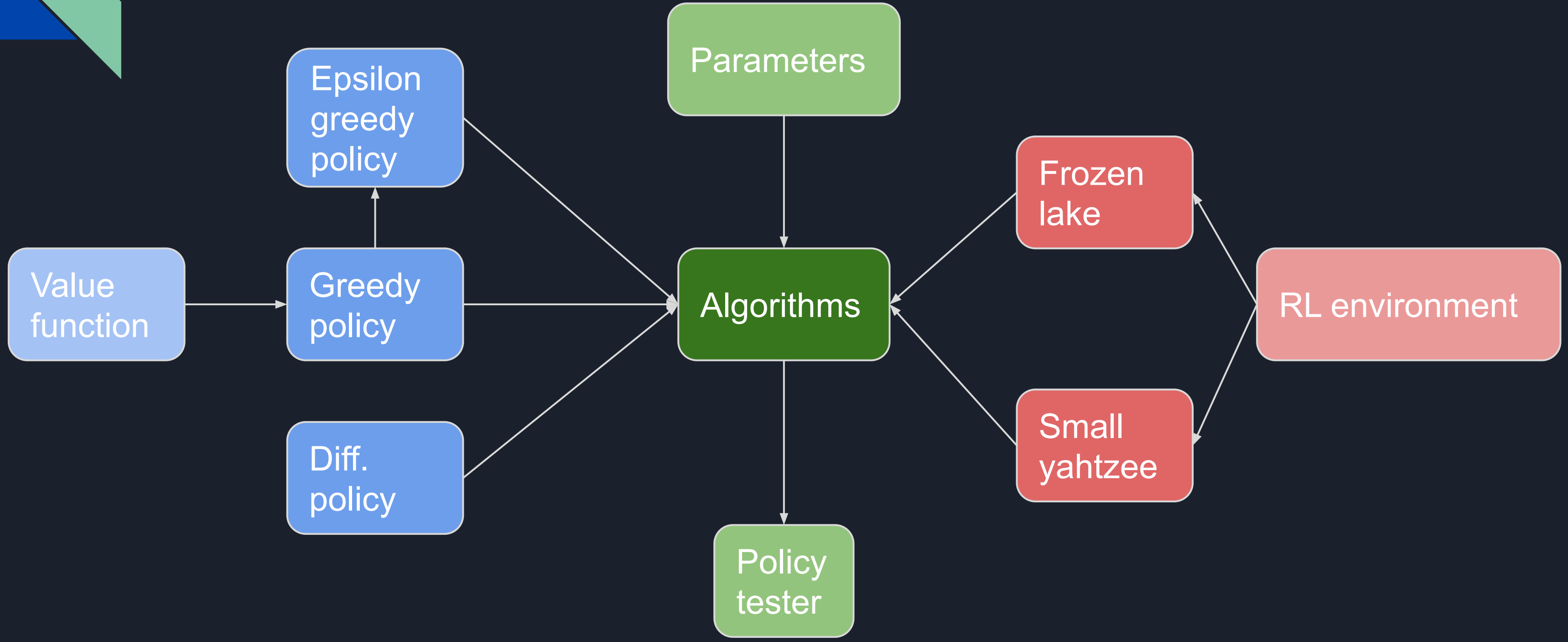
**REINFORCE with Baseline (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$**

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Algorithm parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
  Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} R_k$  $(G_t)$
    $\delta \leftarrow G - \hat{v}(S_t,\mathbf{w})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \gamma^t \delta \nabla \hat{v}(S_t,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \gamma^t \delta \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$

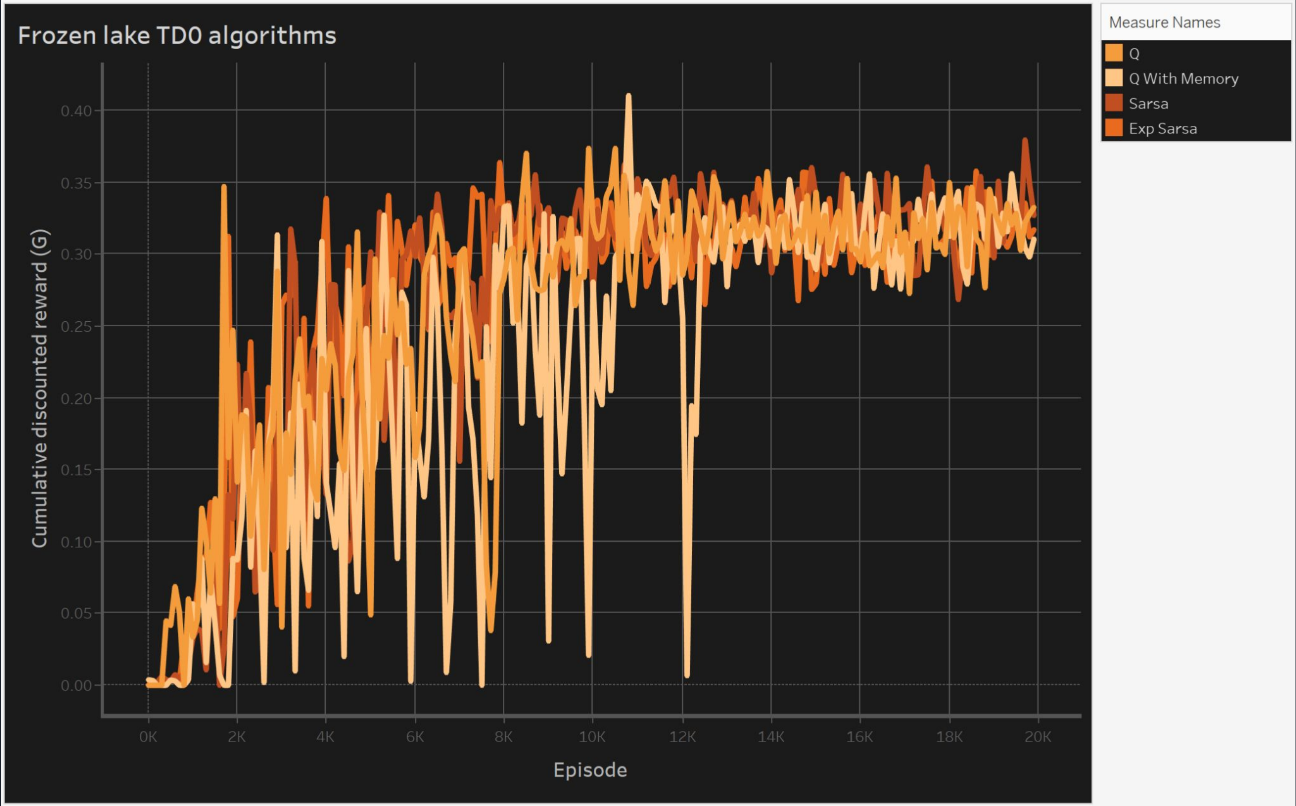
- Remove the “middle-man”
- Faster
- High variance (introduce baseline)

# Implementation:

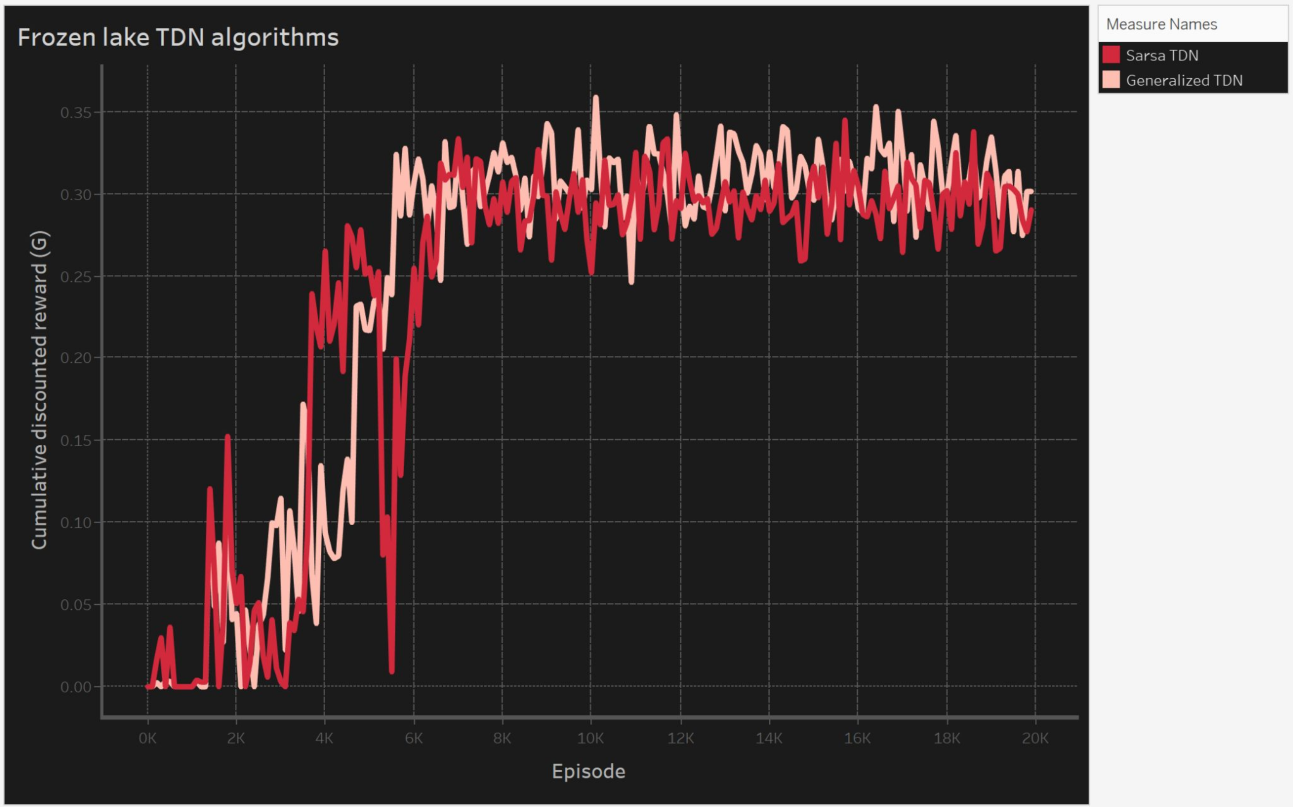


# Implementation:

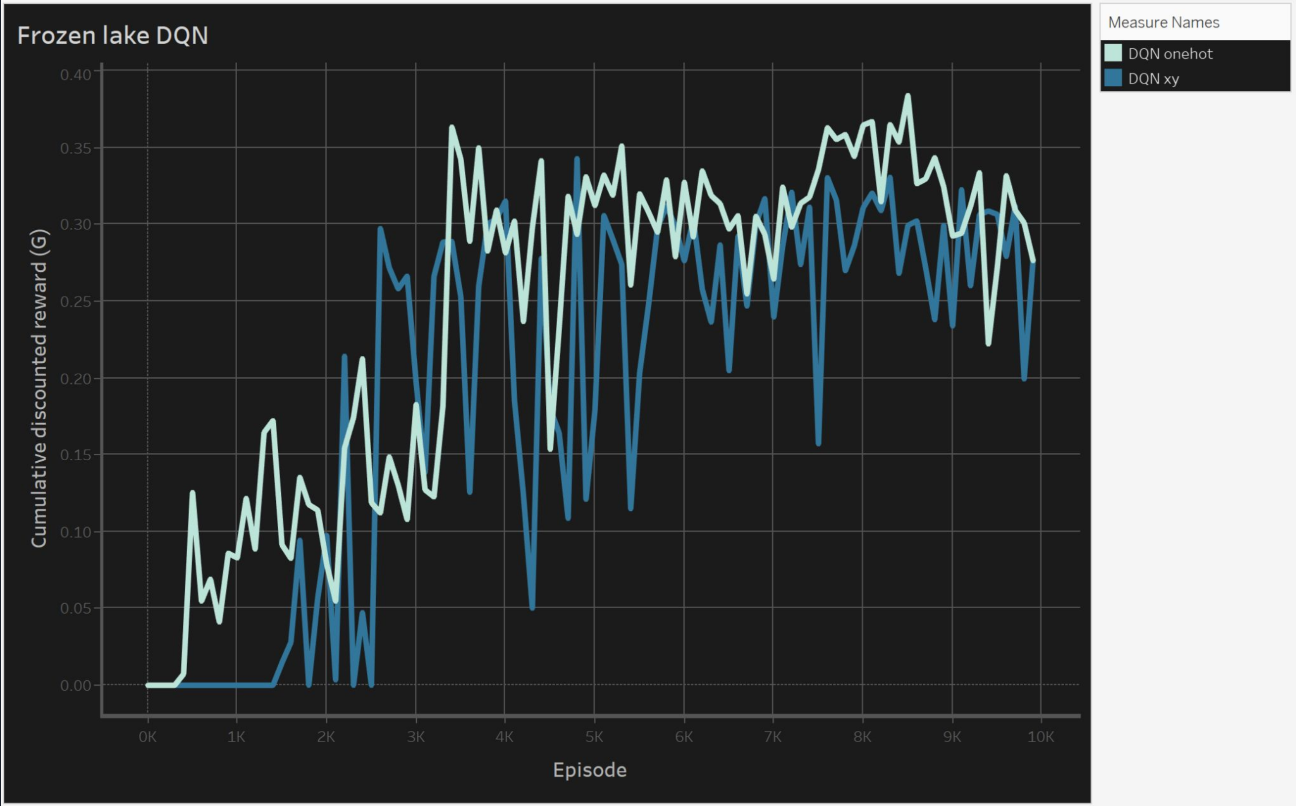
# Results frozen lake



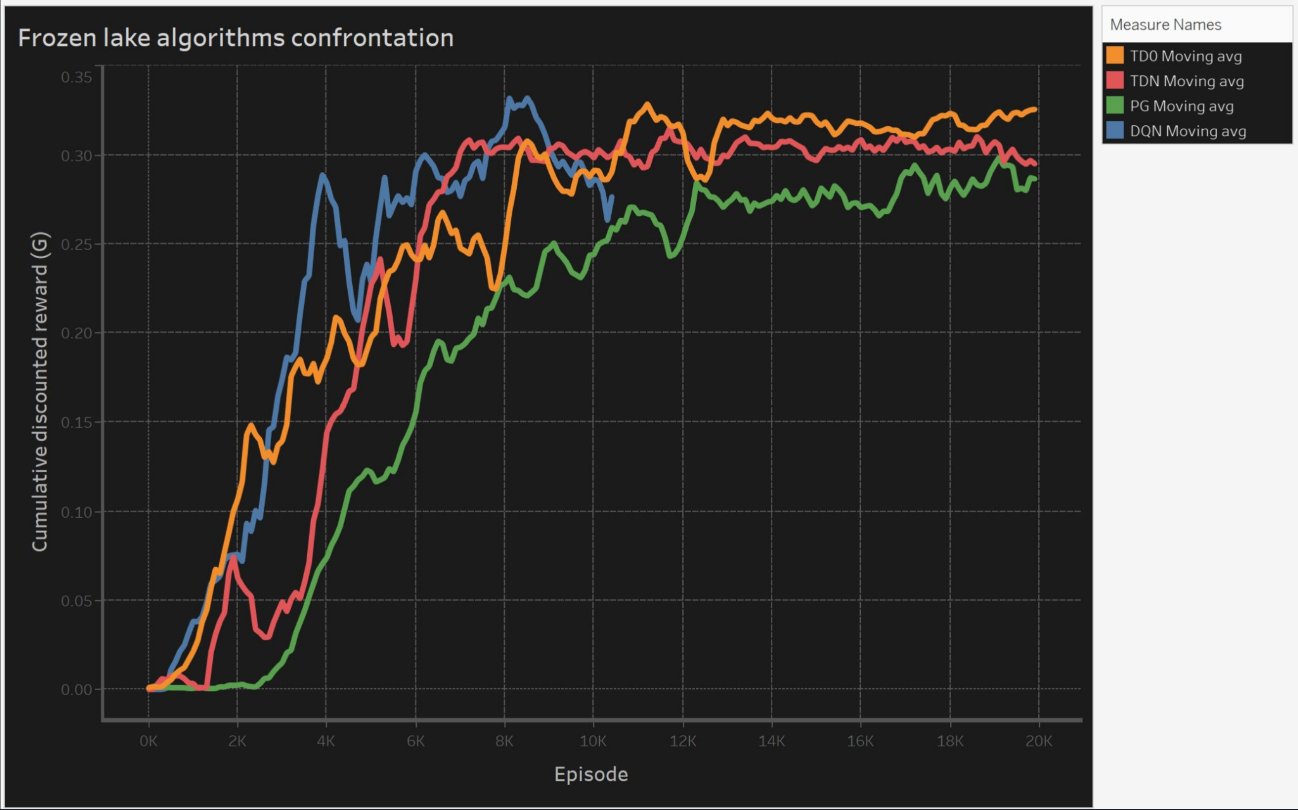
# Results frozen lake



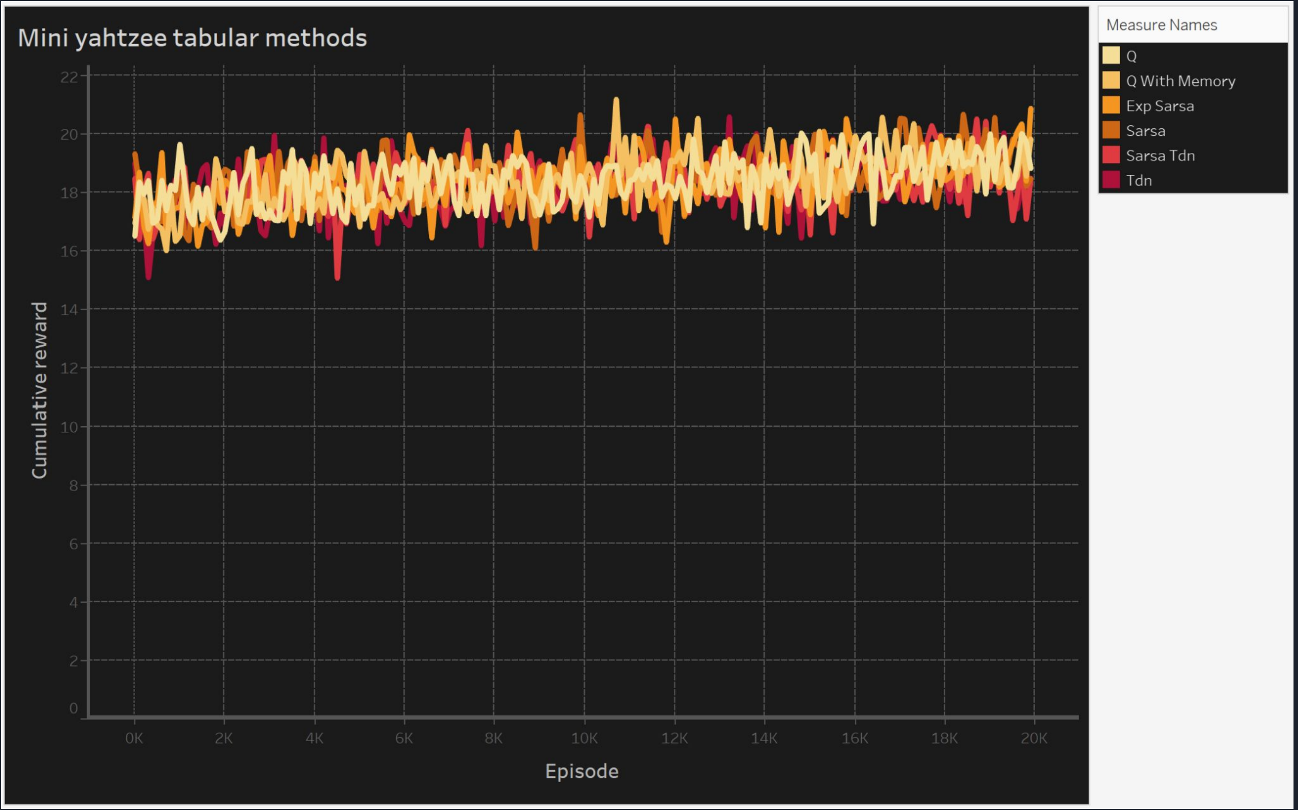
# Results frozen lake



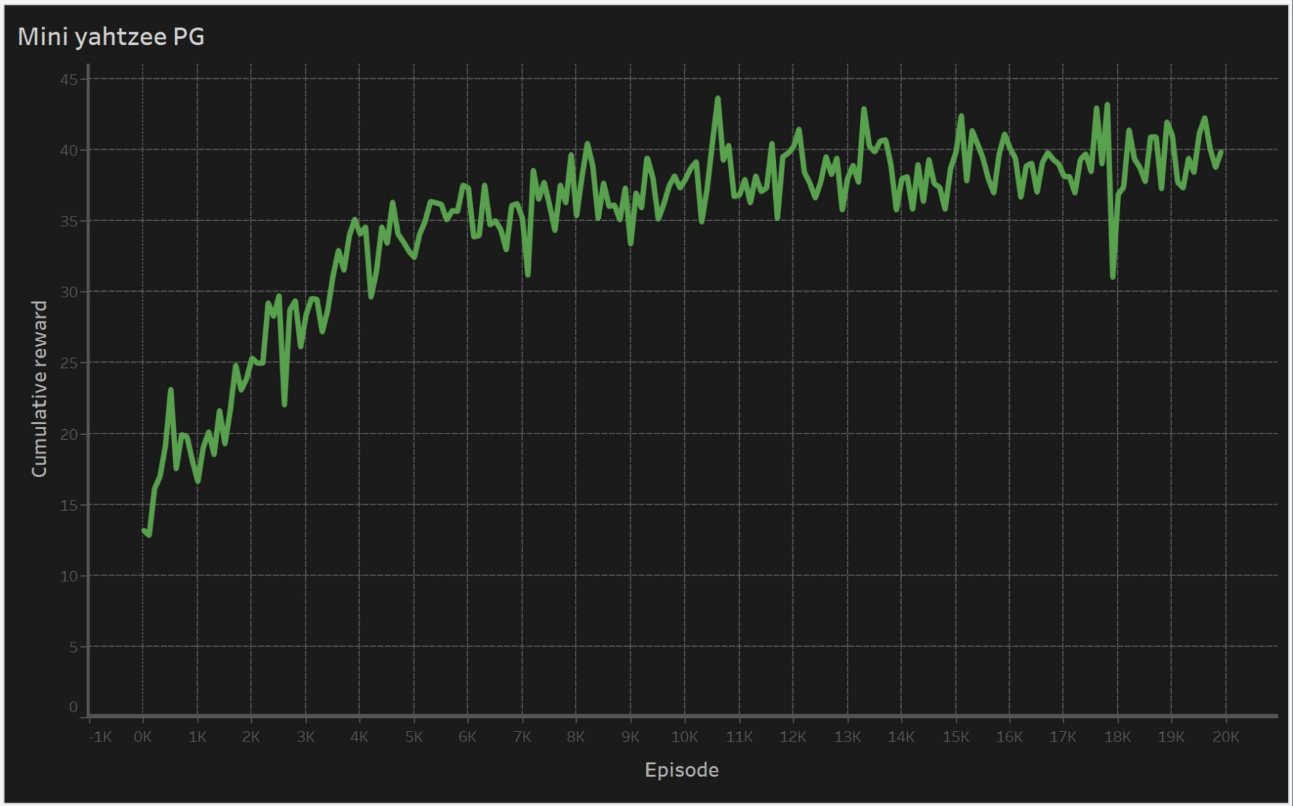
# Results frozen lake



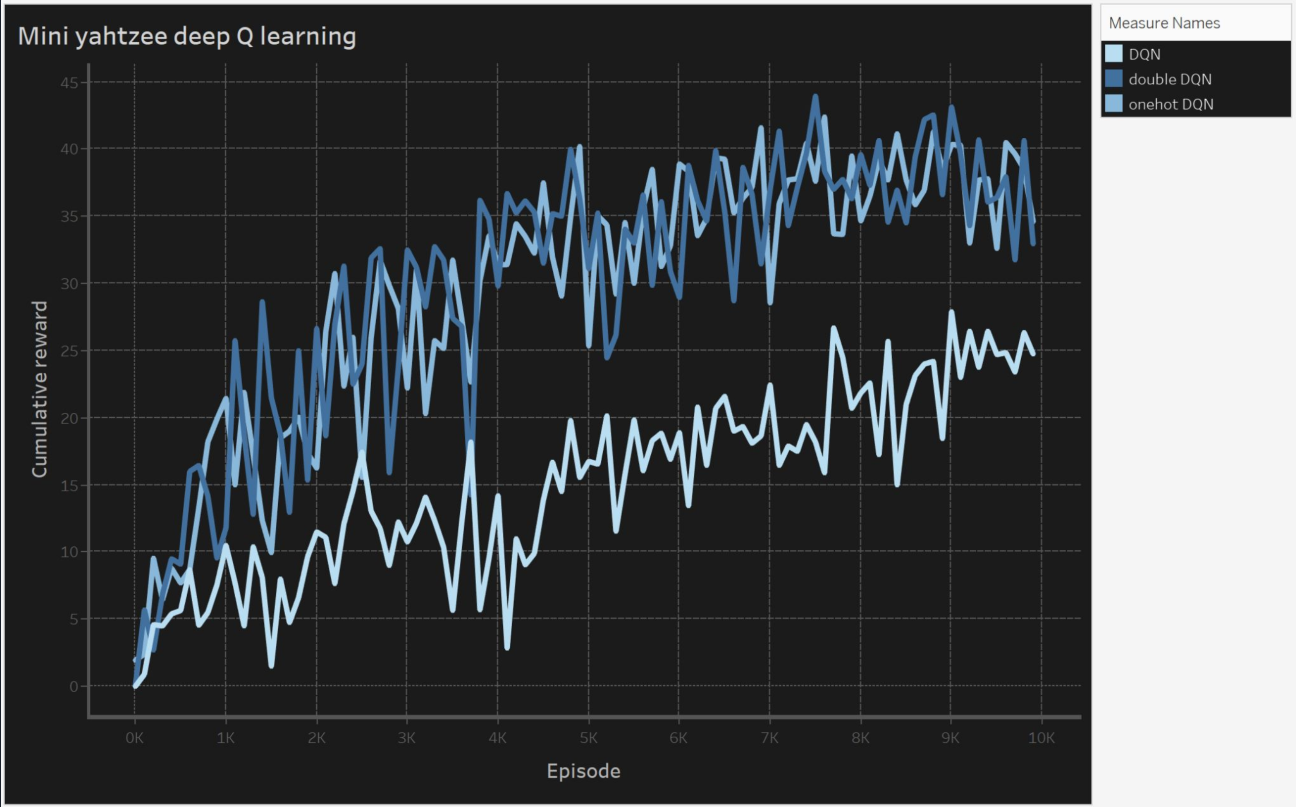
# Results mini yahtzee



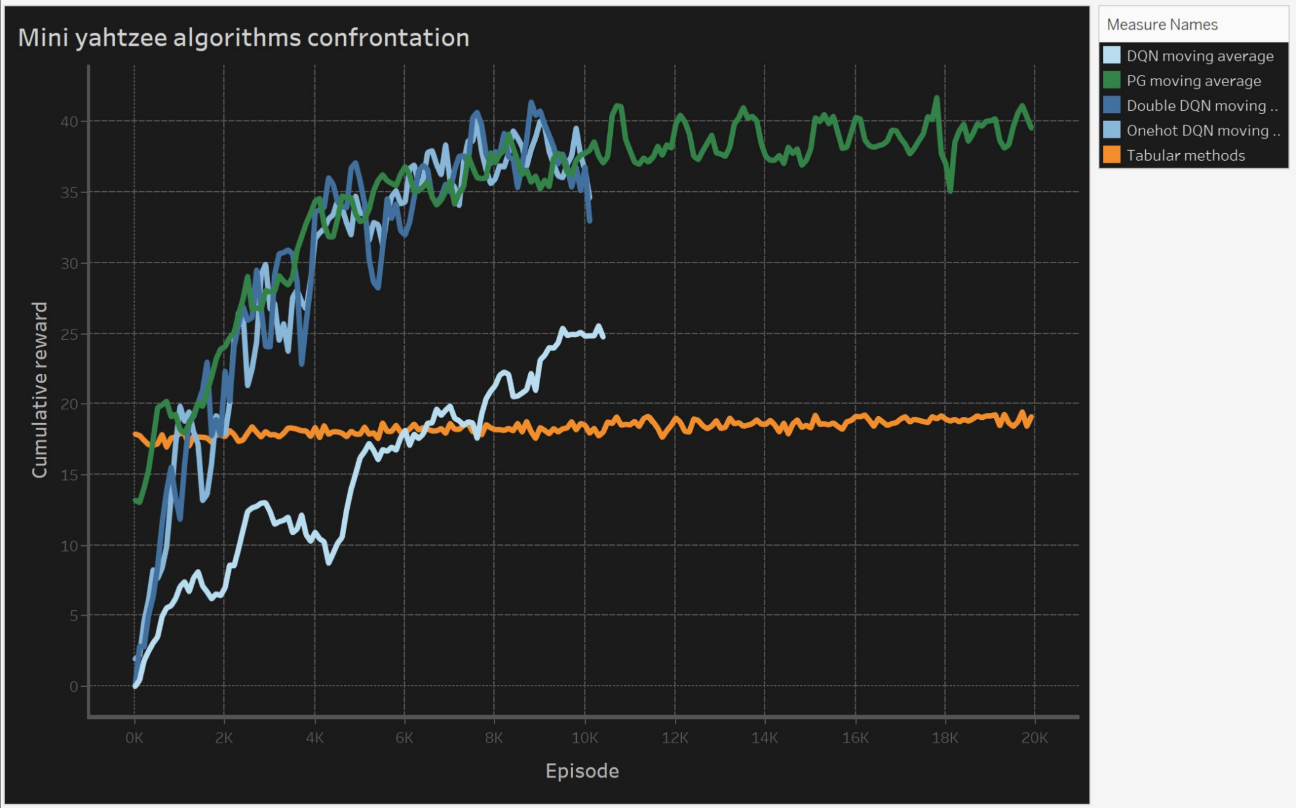
# Results mini yahtzee



# Results mini yahtzee



# Results mini yahtzee



# Conclusion

# Thanks for the attention