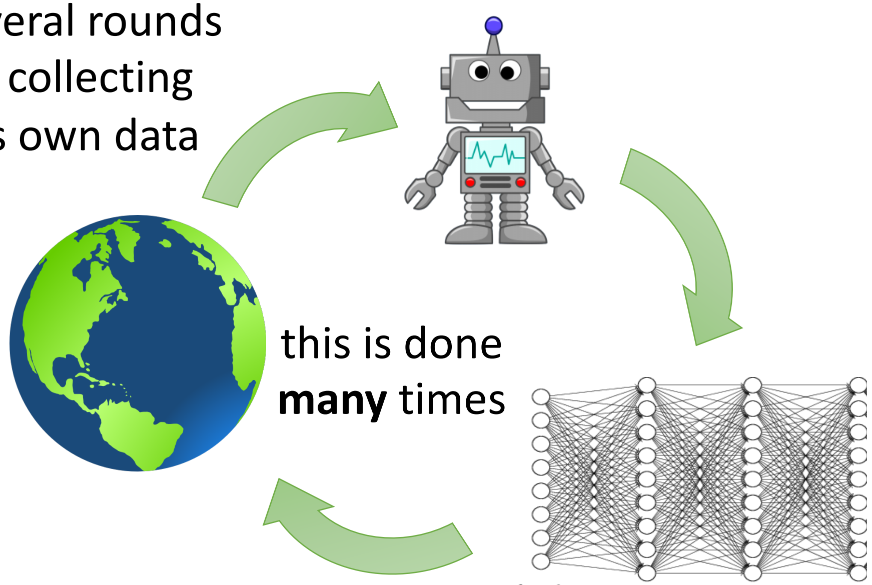
Making Deep RL Easier to Use:

Alleviating Optimization and Tuning Challenges in Deep RL

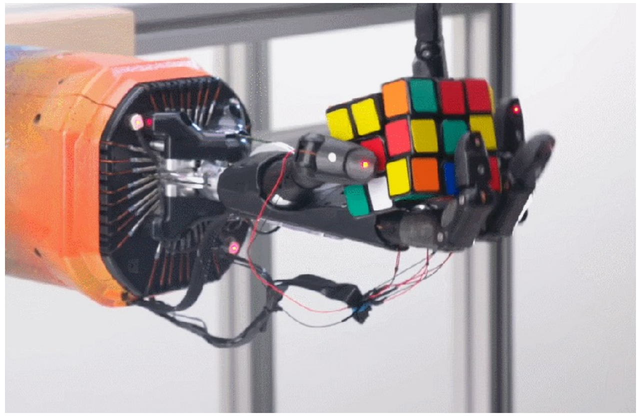
Aviral Kumar
UC Berkeley

Reinforcement Learning

several rounds
of collecting
its own data

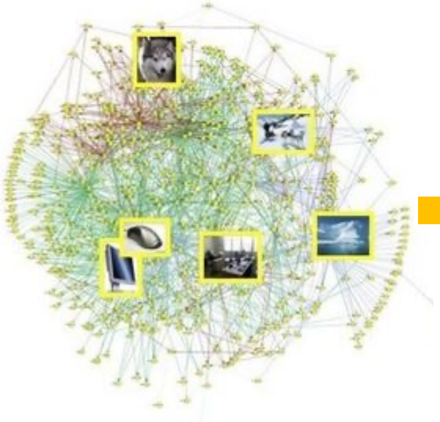


Large amounts of
training time!

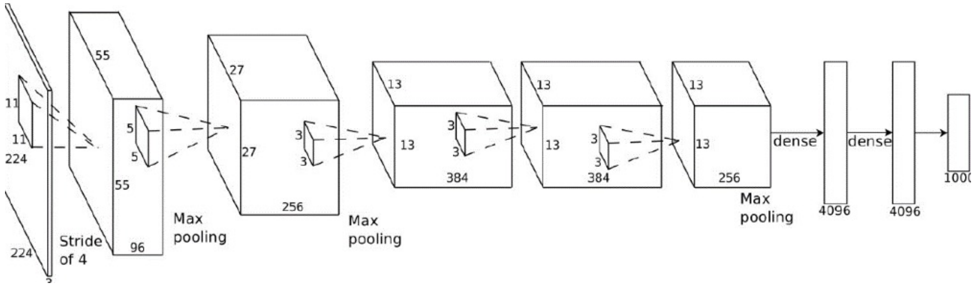


Narrow generalization

This is not like how we do it in
supervised learning, where we use
datasets + large networks

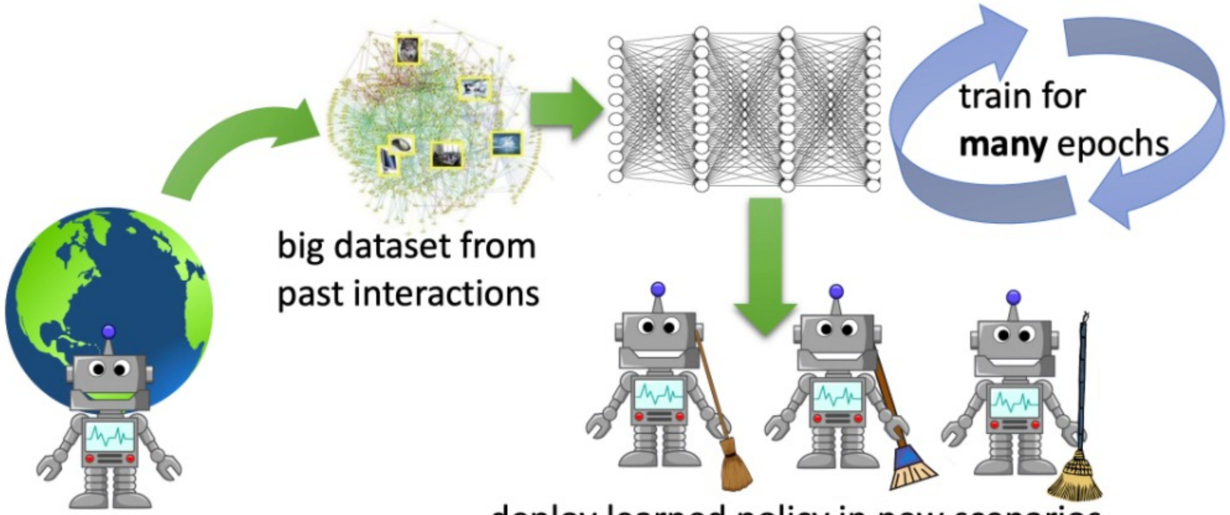


+

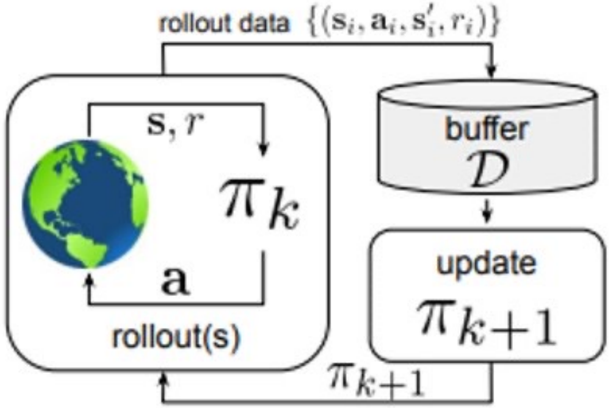


Reinforcement Learning from Static Datasets

Offline RL



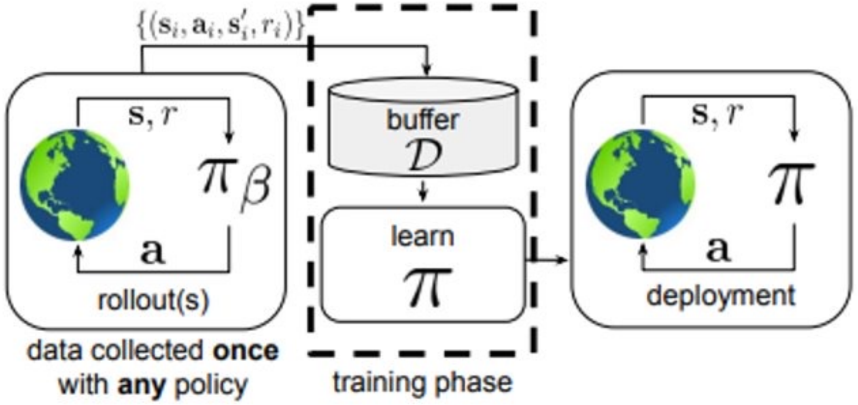
online RL



Better generalization: large networks, diverse datasets

Can do all sorts of cool things: use unlabeled data, task-agnostic data, respect safety constraints, etc.

offline RL

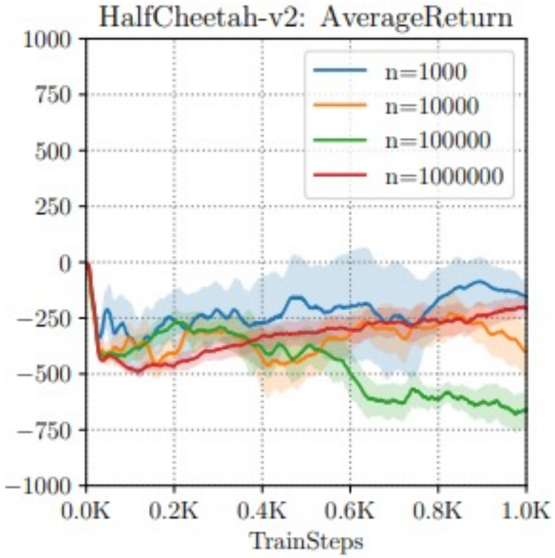


….but is it easy to use?

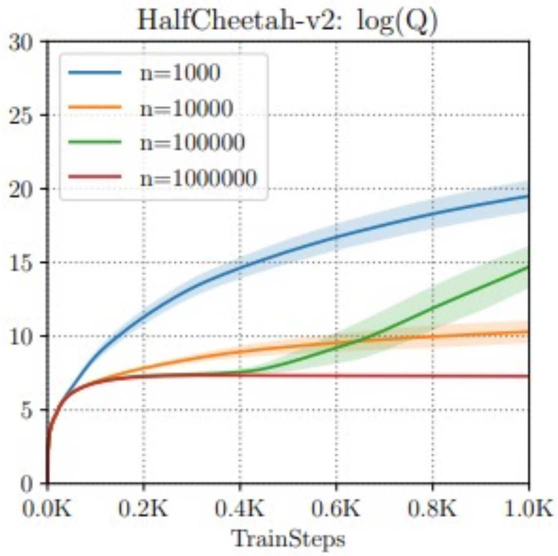
Levine, **K.**, Tucker, Fu. **Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems.** ’20

Challenges in Offline Reinforcement Learning

Challenge 1: Answering counterfactual questions accurately is hard!



how well it does



how well it *thinks* it does (Q-values)

Overestimates the value of unseen outcomes

$$Q(s,a) \leftarrow r(s,a) + \gamma \max_{a'} Q(s',a')$$

$$\boxed{\neq \pi_\beta(a|s)}$$

$$Q(s,a) \leftarrow r(s,a) + \gamma \mathbb{E}_{a' \sim \pi(a'|s')} Q(s',a')$$

$$\boxed{= \pi_\beta(a|s)}$$

Training: $\mathbb{E}_{s,a \sim d^{\pi_\beta}(s,a)} \left[\left(Q(s,a) - \mathcal{B}\bar{Q}(s,a) \right)^2 \right]$

Can we solve this distributional shift issue?

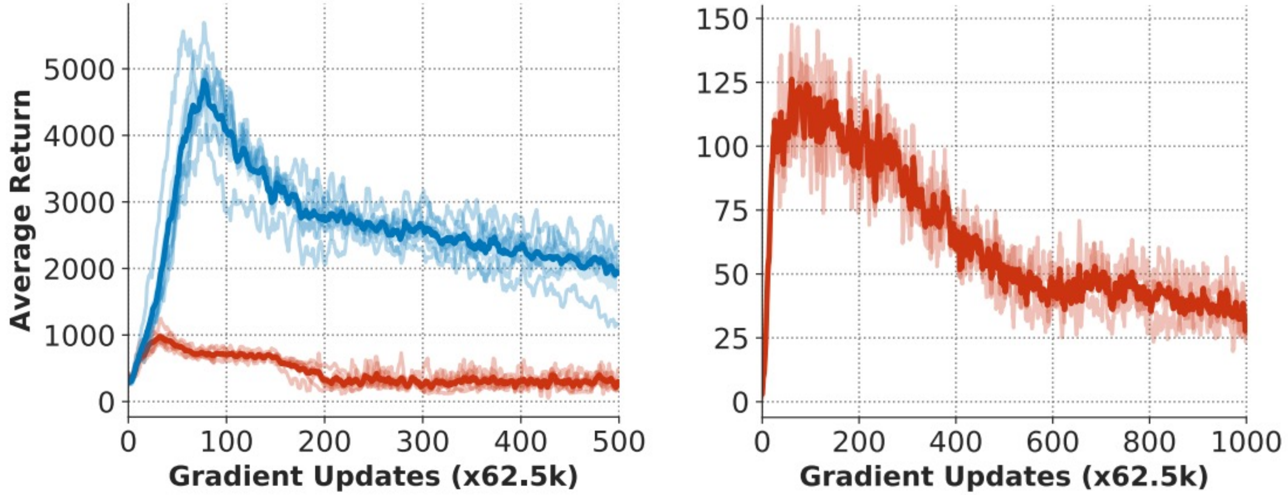
Yes, several algorithms:

1. Algorithms that learn lower-bounds on Q-values
2. Algorithms that constrain behavior close to the data

K., Levine. **NeurIPS Tutorial on Offline Reinforcement Learning.** 2020.

Challenges in Offline Reinforcement Learning

Challenge 2: Issues with optimization and tuning



Performance goes up and comes back down

Learning can be unstable: error may go up with more training

What are the issues? How can we detect and address them?

Supervised learning:
Track train and validation error,
Perform early stopping if overfitting,
Increase network capacity if underfitting

Reinforcement learning:
What to track?
When is the algorithm “overfitting”?
What regularization to add?
Does the algorithm ”underfit”, but it appears as “overfitting”?

Understanding Optimization Challenges in RL

Bellman equation

$$Q^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \leftarrow r(\mathbf{s}_t, \mathbf{a}_t) + \gamma E[Q^{\pi_\theta}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})]$$

Q-Learning

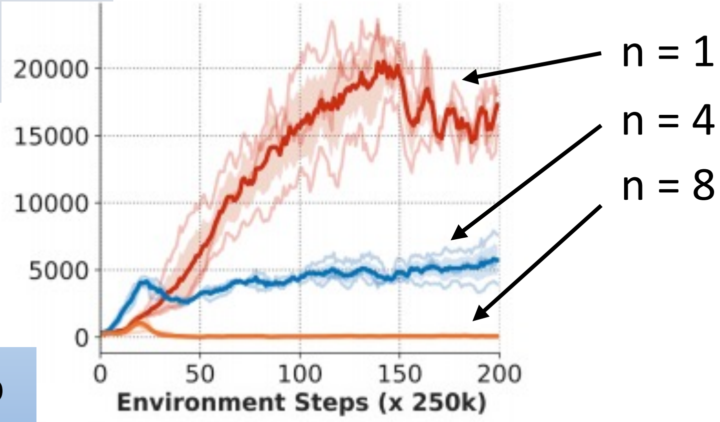
1. Train Q-functions by minimizing TD Error:

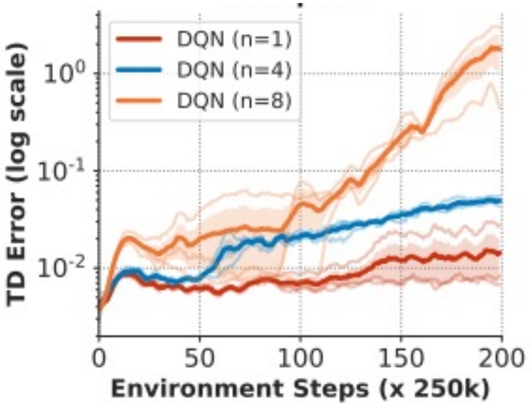
$$E_{(\mathbf{s},\mathbf{a})\sim\pi_\beta(\mathbf{s},\mathbf{a})}\left[(Q_\phi(\mathbf{s},\mathbf{a}) - (r(\mathbf{s},\mathbf{a}) + \gamma E[Q_\phi(\mathbf{s}',\mathbf{a}')]))^2\right]$$

2. (Optional) Collect new data in the environment by rolling out the learned policy

Gradient descent

Training >=1 step per datapoint leads to poor performance





OK, maybe I need to prevent overfitting?

But training error is high with larger n

K.*, Agarwal*, Ghosh, Levine. **Implicit Under-Parameterization Inhibits Data-Efficient Deep RL.** ICLR 2021

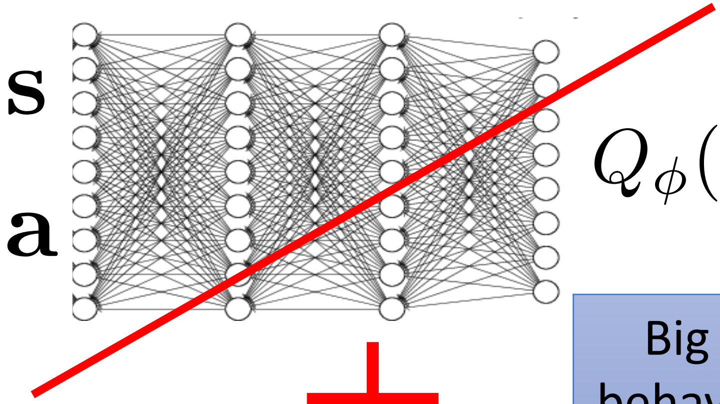
Implicit Under-Parameterization

TD Error (Regressing to itself)

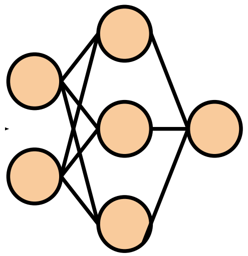
$$E_{(\mathbf{s},\mathbf{a})\sim\pi_\beta(\mathbf{s},\mathbf{a})}\left[\left(Q_\phi(\mathbf{s},\mathbf{a}) - (r(\mathbf{s},\mathbf{a}) + \gamma E[Q_\phi(\mathbf{s}',\mathbf{a}')])\right)^2\right]$$

$$Q_\phi(\mathbf{s},\mathbf{a}) = \mathbf{w}^T\Phi_\phi(\mathbf{s},\mathbf{a}) \quad \Phi_\phi(\mathbf{s},\mathbf{a}) \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|\times d}$$

Learned features



$Q_\phi(\mathbf{s},\mathbf{a})$



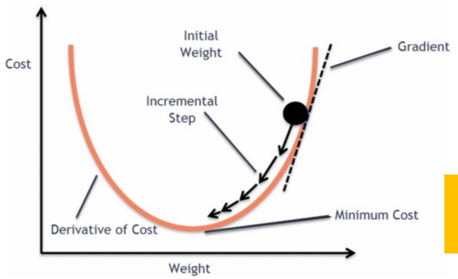
Big network **implicitly** behaves as a low-capacity, **under-parameterized** network

IUP = Feature rank collapse

Rank collapse

More aliasing

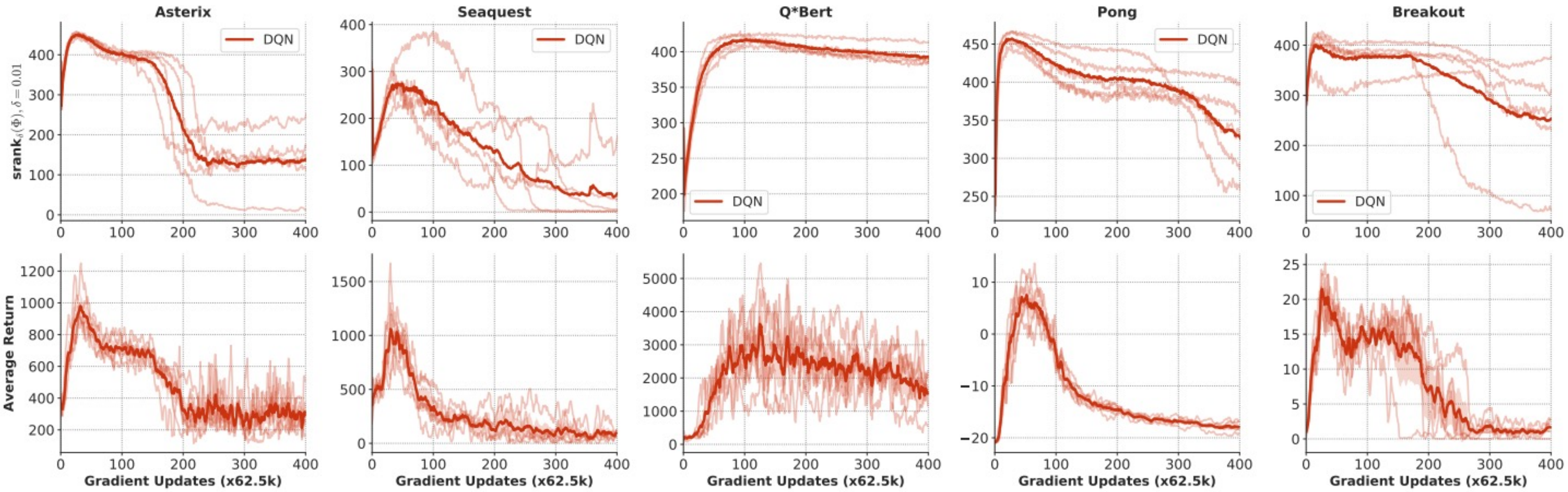
Poor performance



Gradient descent optimizer

K.*, Agarwal*, Ghosh, Levine. **Implicit Under-Parameterization Inhibits Data-Efficient Deep RL.** ICLR 2021

Empirical Evidence of Rank Collapse



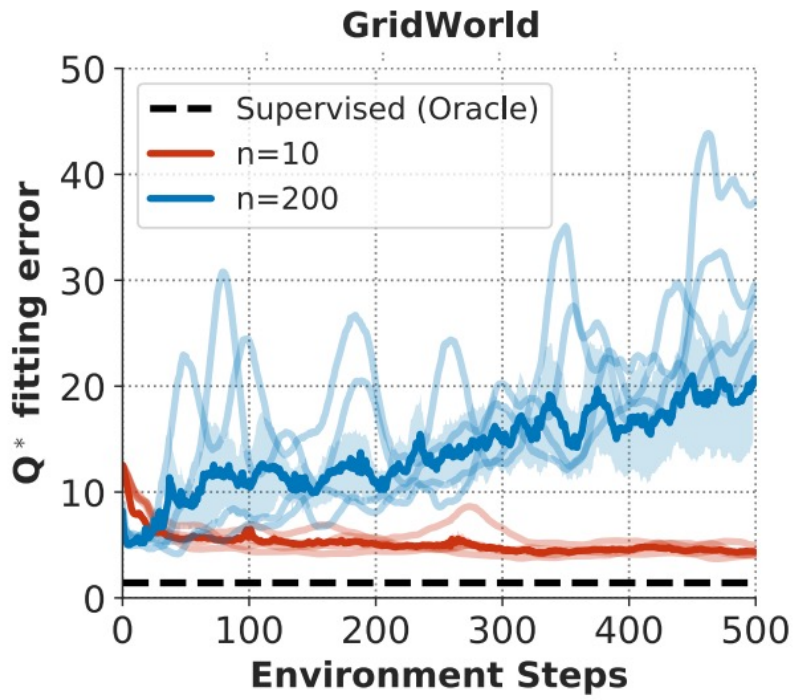
Rank collapse also strongly corresponds to poor performance!

Also, tells us that RL algorithms learn poor representations!

K.*, Agarwal*, Ghosh, Levine. **Implicit Under-Parameterization Inhibits Data-Efficient Deep RL.** ICLR 2021

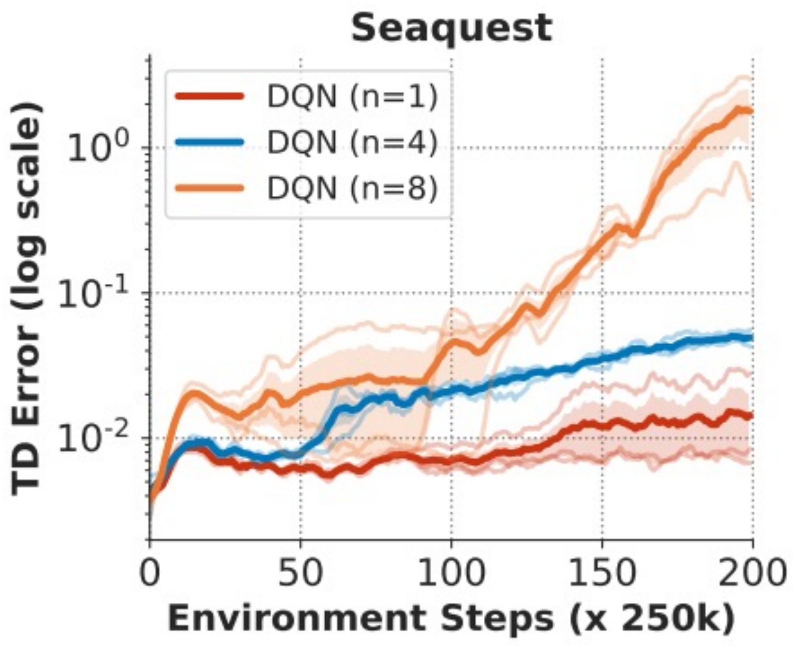
Why is Implicit Under-Parameterization Bad?

Rank collapse inhibits the ability to represent the optimal Q-function



“Aliasing” effects

Also leads to increased training TD errors in several cases

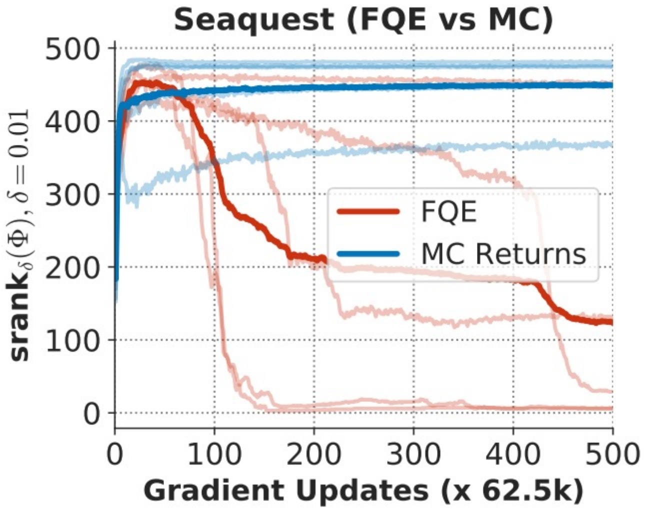


Lack of expressivity to minimize training loss

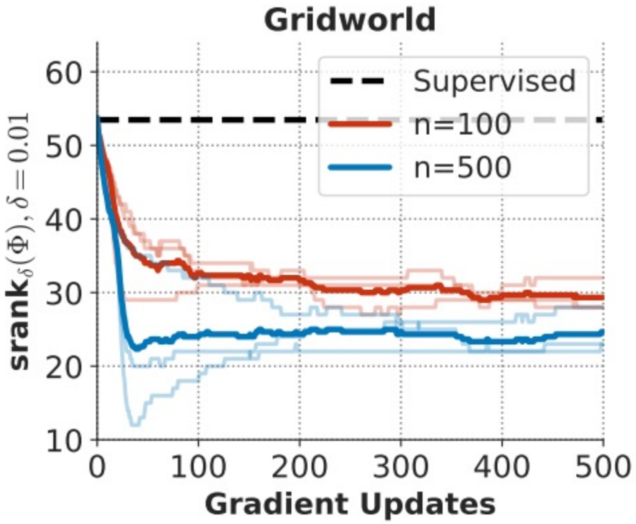
K.*, Agarwal*, Ghosh, Levine. **Implicit Under-Parameterization Inhibits Data-Efficient Deep RL.** ICLR 2021

What Causes Implicit Under-Parameterization?

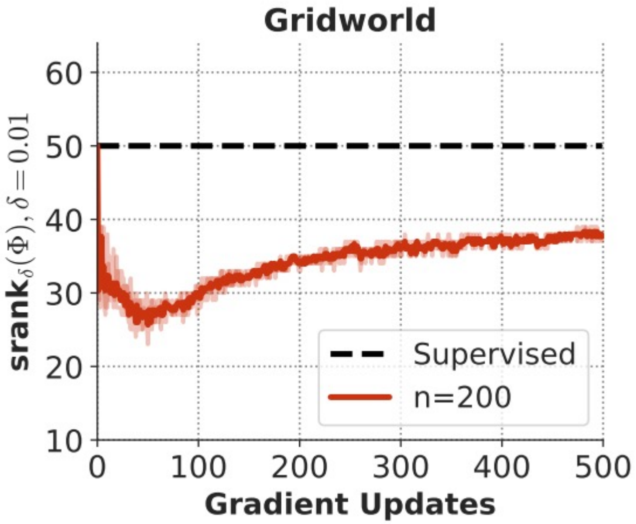
Rank decreases only with bootstrapping



"Moving" nature of objective is not the cause



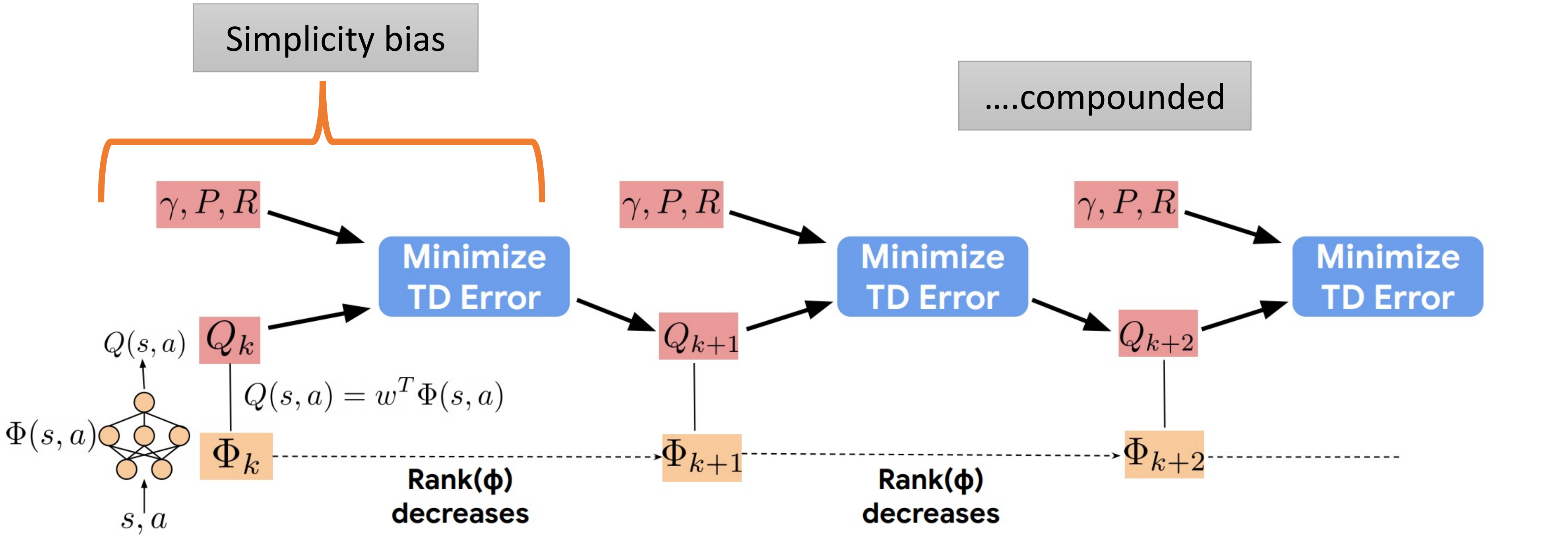
Does not arise with "oracle" Q-target values



- Does not happen without bootstrapping
- Moving objectives is not the issue, but something specific to bootstrapping is
- Using oracle target-values does not lead to rank collapse

K.*, Agarwal*, Ghosh, Levine. **Implicit Under-Parameterization Inhibits Data-Efficient Deep RL.** ICLR 2021

What Causes Implicit Under-Parameterization?



Standard TD error

$$\mathbf{Q}_{k+1} \leftarrow \arg\min_{\mathbf{Q} \in \mathcal{Q}} \sum_{\mathbf{s}_i, \mathbf{a}_i \in \mathcal{D}} \left(Q(\mathbf{s}_i, \mathbf{a}_i) - y_k(\mathbf{s}_i, \mathbf{a}_i)\right)^2$$

Implicit regularizer

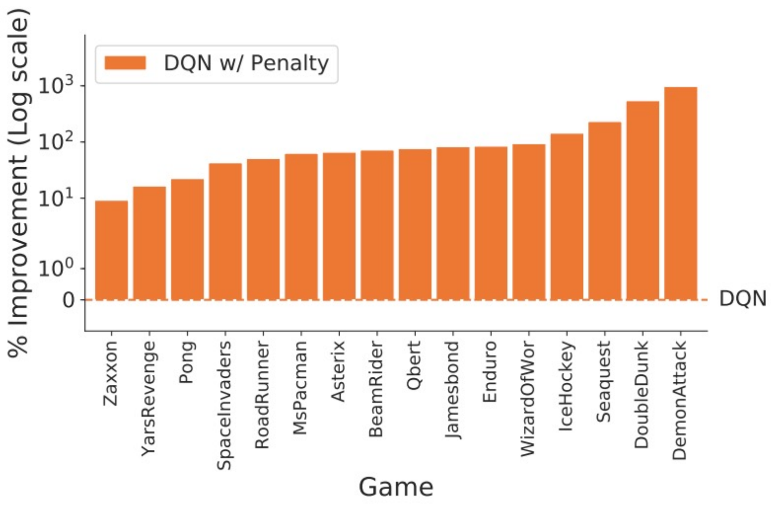
K.*, Agarwal*, Ghosh, Levine. **Implicit Under-Parameterization Inhibits Data-Efficient Deep RL.** ICLR 2021

How Can We Fix Implicit Under-Parameterization?

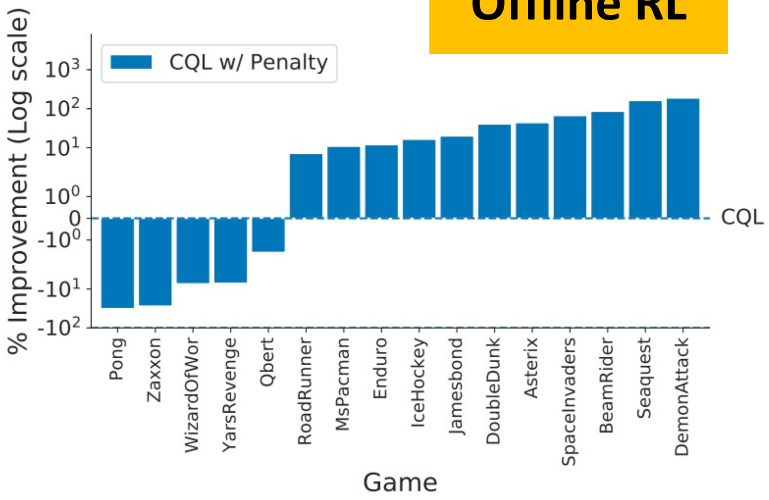
Take 1: Address the symptom

$$\mathcal{L}_p(\Phi) = \sigma^2_{\max}(\Phi) - \sigma^2_{\min}(\Phi).$$

Intuition: Penalize the disproportionate increase in singular values



Offline RL

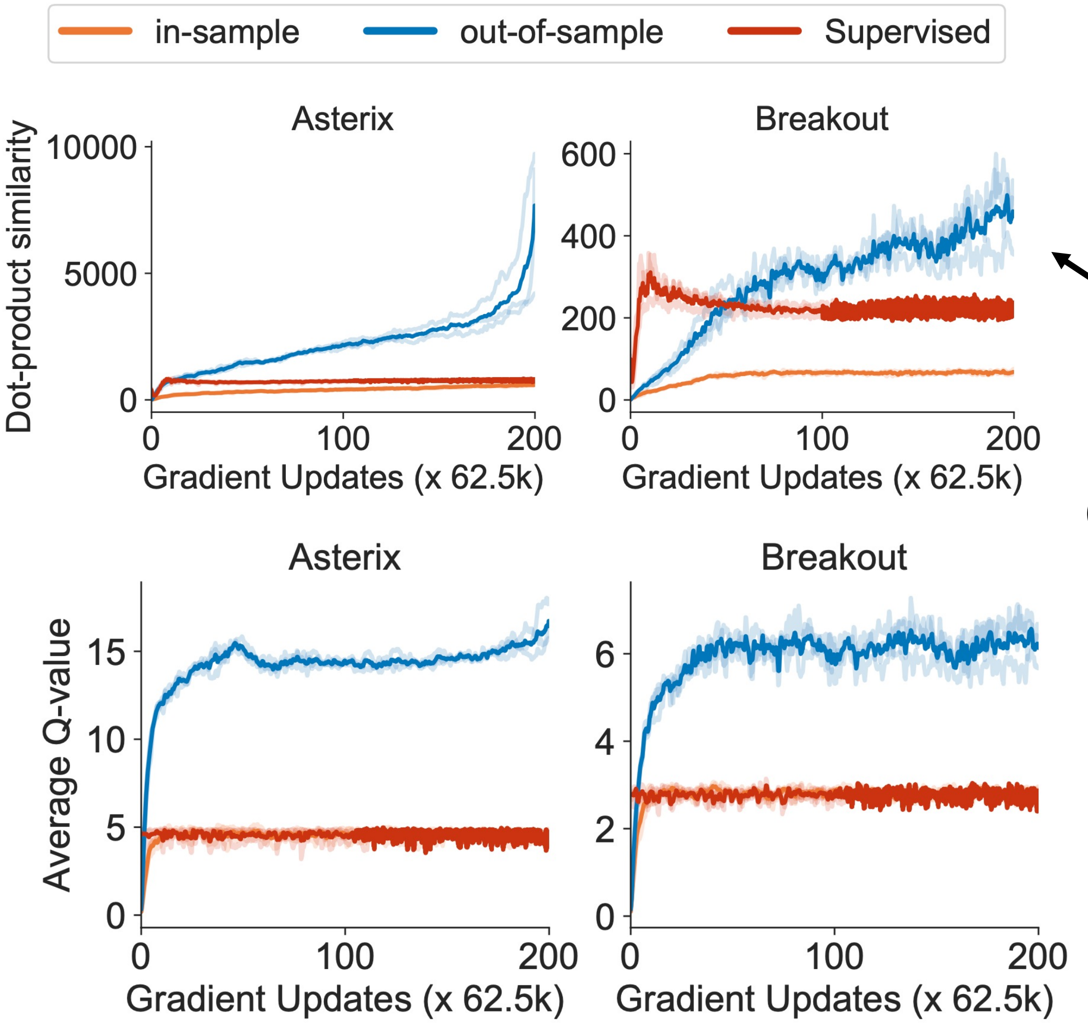


But this doesn't seem fundamental, since optimal solutions may not have highest rank.

Can we directly address the cause of the issue?

K.*, Agarwal*, Ghosh, Levine. **Implicit Under-Parameterization Inhibits Data-Efficient Deep RL.** ICLR 2021

Take 2: Understanding Optimization Dynamics

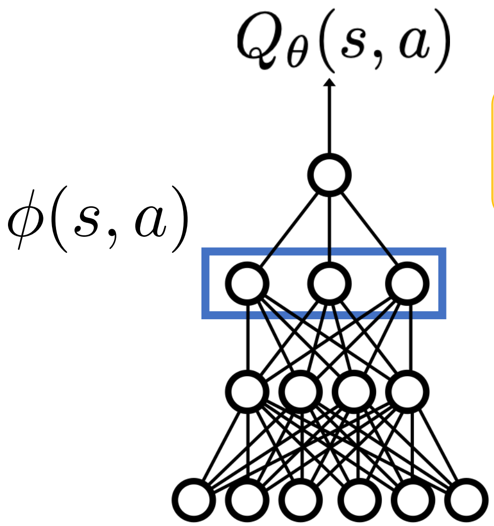


$$\mathcal{D} = \{(s_i, a_i, r_i, s_i')\}_{i=1}^{N}$$

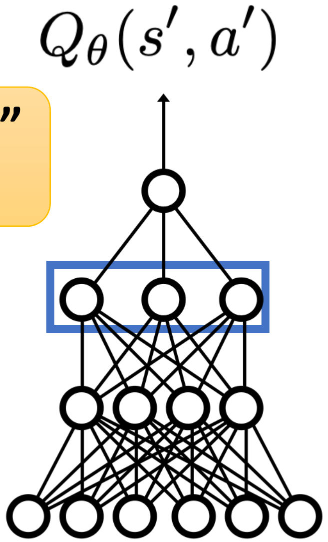
$$Q(s,a) \leftarrow r(s,a) + \gamma Q(s', a')$$

$$\sum_{(s,a,s',a')} \phi(s,a)^\top \phi(s',a')$$

Feature dot products increase for unseen actions in the backup

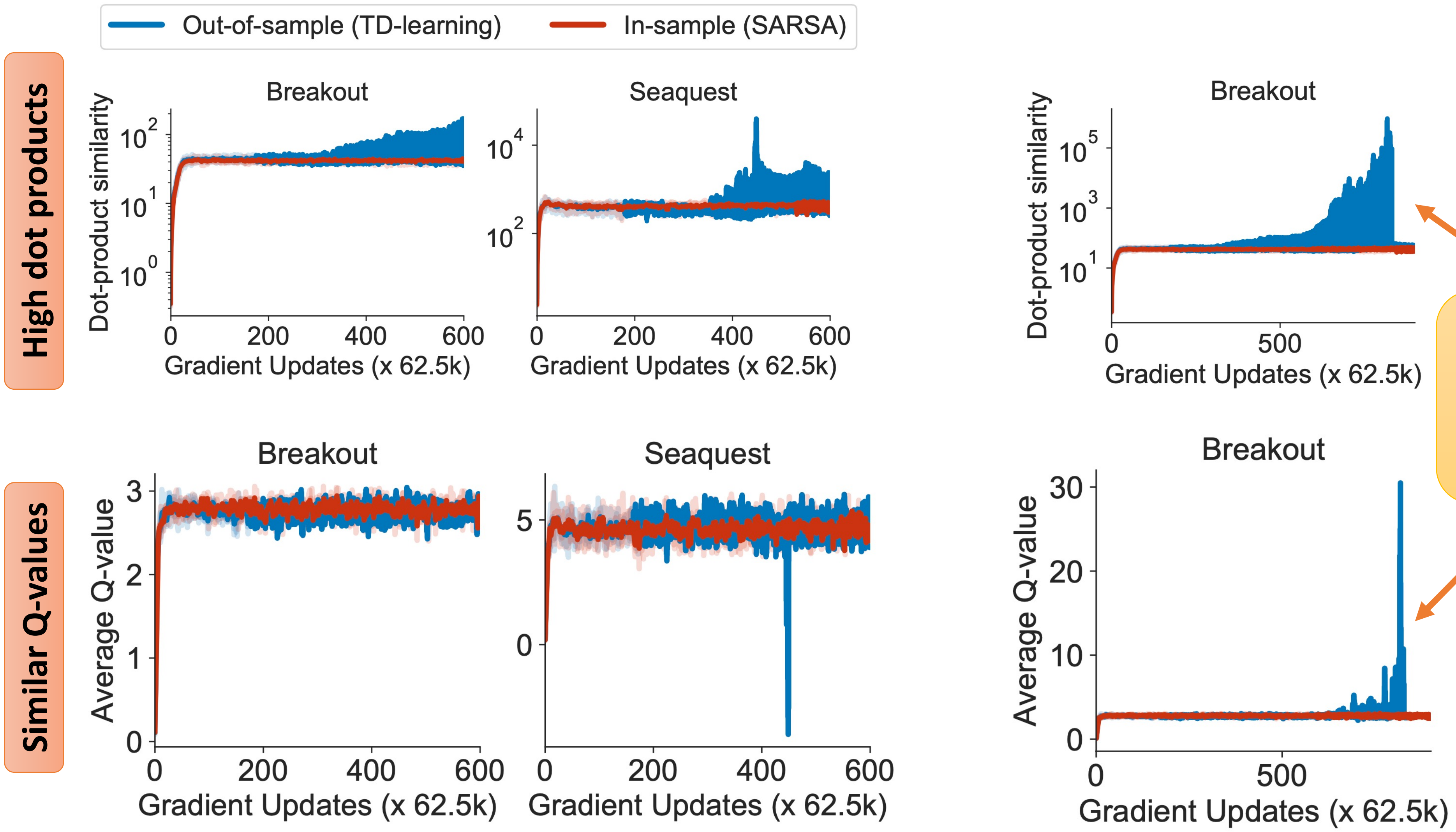


“Co-adapted” features



K. et al. **Value-Based Deep RL Requires Explicit Regularization.** In preparation, 2021.

Feature Dot Products Increase over Training

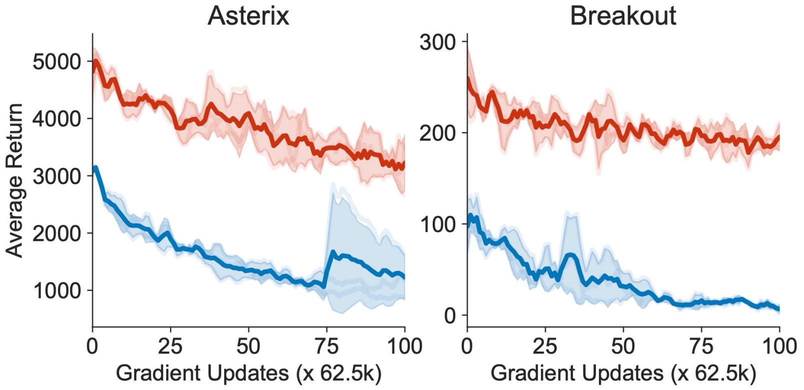


Incorrect Q-values learned eventually; dot-products increase throughout training

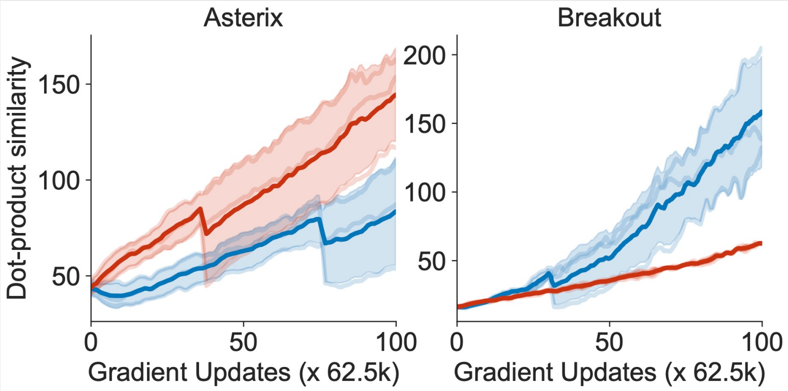
K. et al. **Value-Based Deep RL Requires Explicit Regularization.** In preparation, 2021.

High Feature Dot Products and Performance

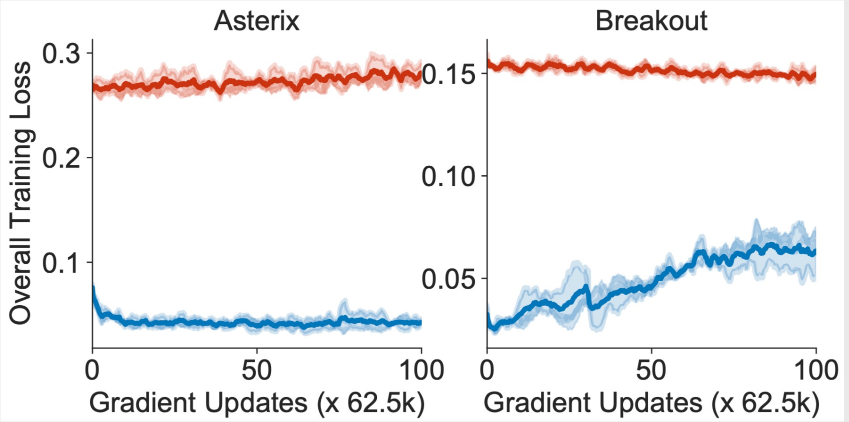
Return degrades from good solutions



Dot products increase, returns decrease



Overall training error is still low!



An **implicit regularization** phenomenon that arises when running gradient descent with bootstrapping, that leads us to maximize dot products of features

$$\sum_{(s,a,s',a')} \phi(s,a)^\top \phi(s',a')$$

K. et al. **Value-Based Deep RL Requires Explicit Regularization.** In preparation, 2021.

How can we reduce this implicit regularization?

Can we add an explicit regularizer to convert this deep RL implicit regularizer to that in SL?

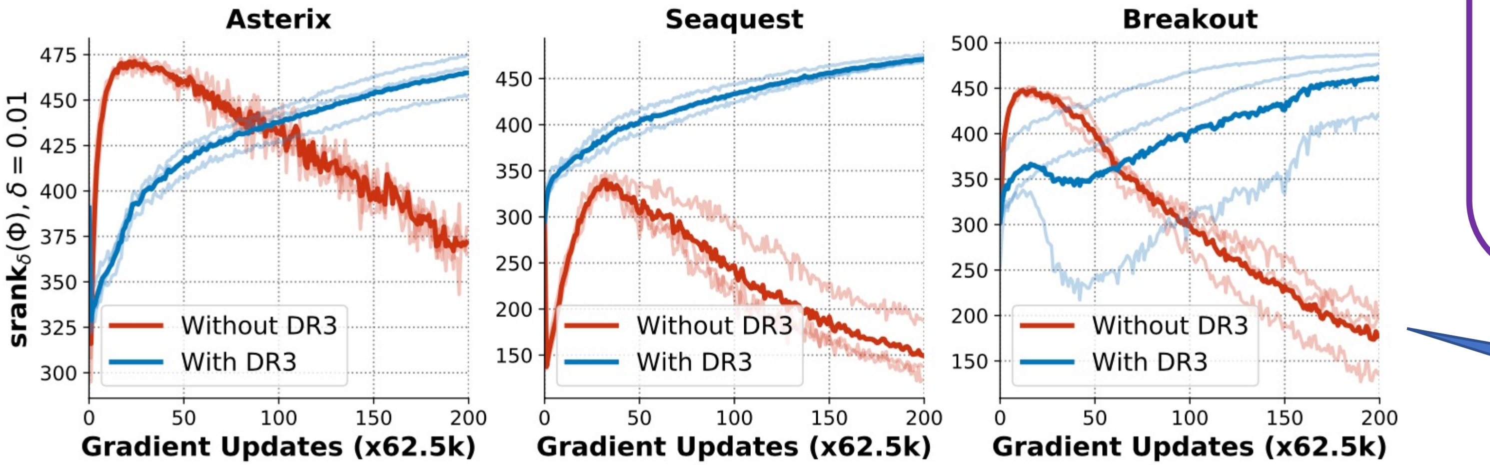
$$\mathcal{R}_{\mathrm{SL}}(\phi) = \sum_{(s,a)} \phi(s,a)^\top \phi(s,a)$$

$$\mathcal{R}_{\mathrm{RL}}(\phi) = \sum_{(s,a,s',a')} \phi(s,a)^\top \phi(s',a') - \gamma \phi(s,a)^\top \phi(s',a')$$

Instead add this back in explicitly!

Our Method (DR3): Add an explicit regularizer that minimizes dot products

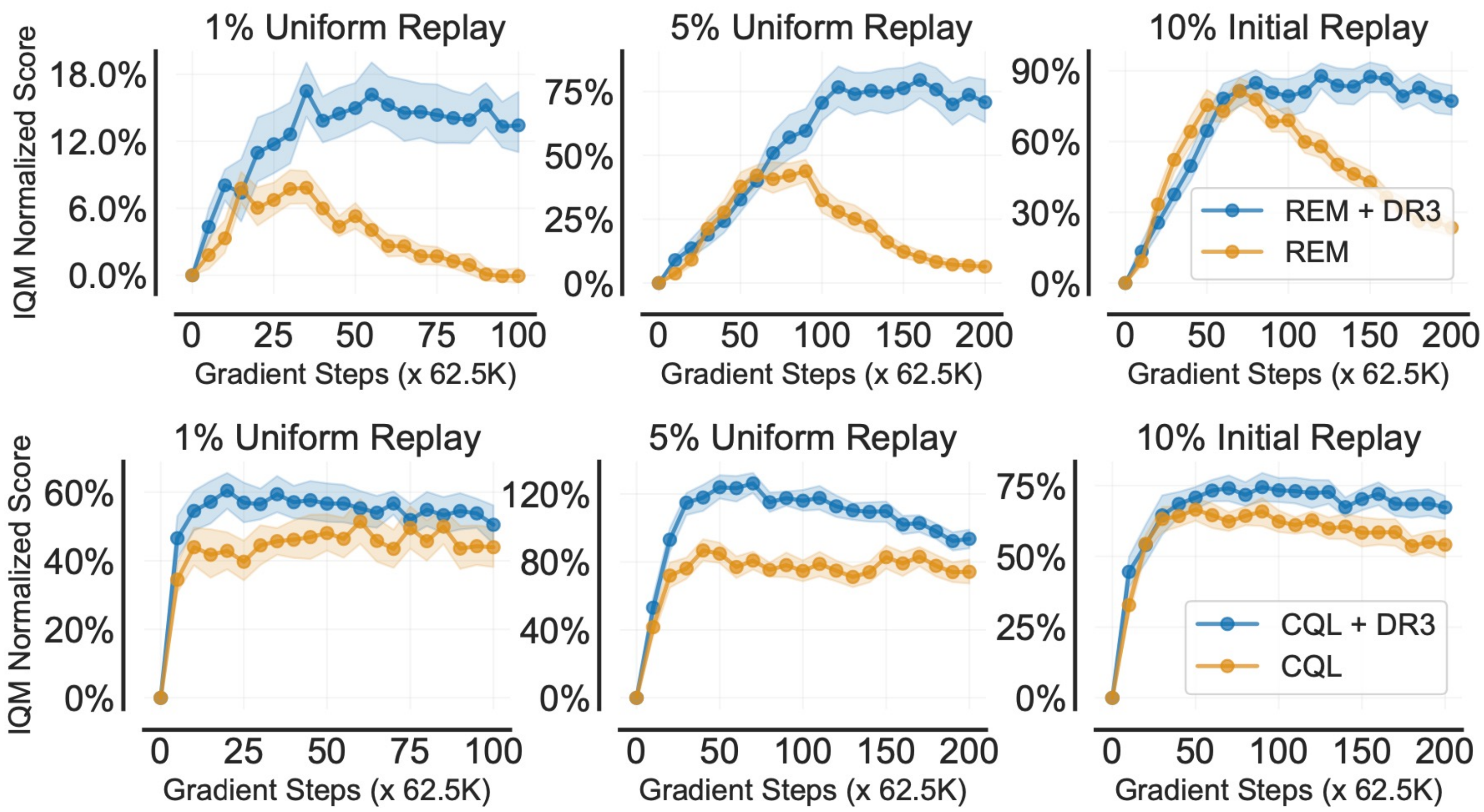
$$\mathcal{R}_{\exp}(\phi) = \sum_{(s,a,s',a')} \phi(s,a)^\top \phi(s',a')$$



Also mitigates rank collapse!

K. et al. **Value-Based Deep RL Requires Explicit Regularization.** In preparation, 2021.

Empirical Performance on Offline RL Benchmarks



17 Atari games

2 base offline RL methods

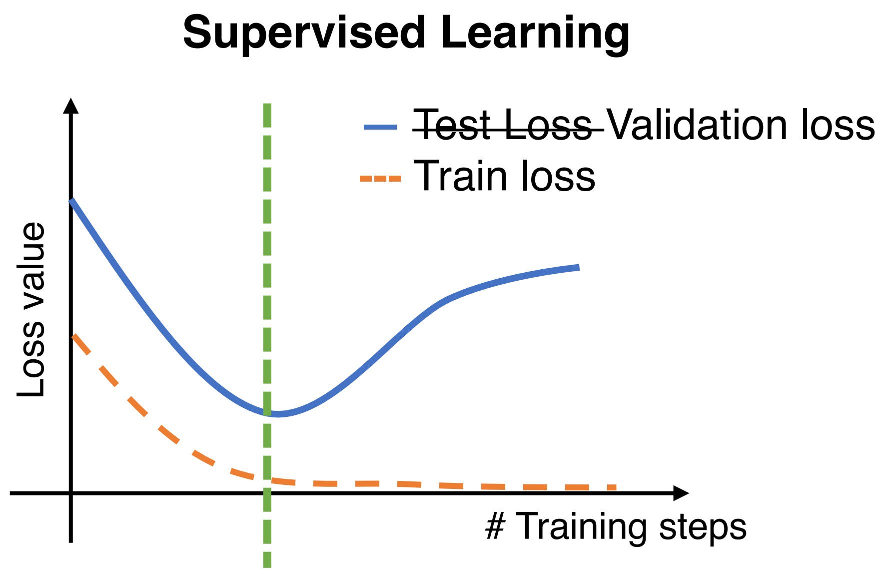
Improves both stability and performance

Stability is very important

K. et al. **Value-Based Deep RL Requires Explicit Regularization.** In preparation, 2021.

How can these algorithmic advances guide practitioners in debugging and tuning RL algorithms on new applications?

A *Workflow* for Offline Deep RL



RL?

- ❑ Test error different from training objective
- ❑ Distribution of resulting policy different from training dataset

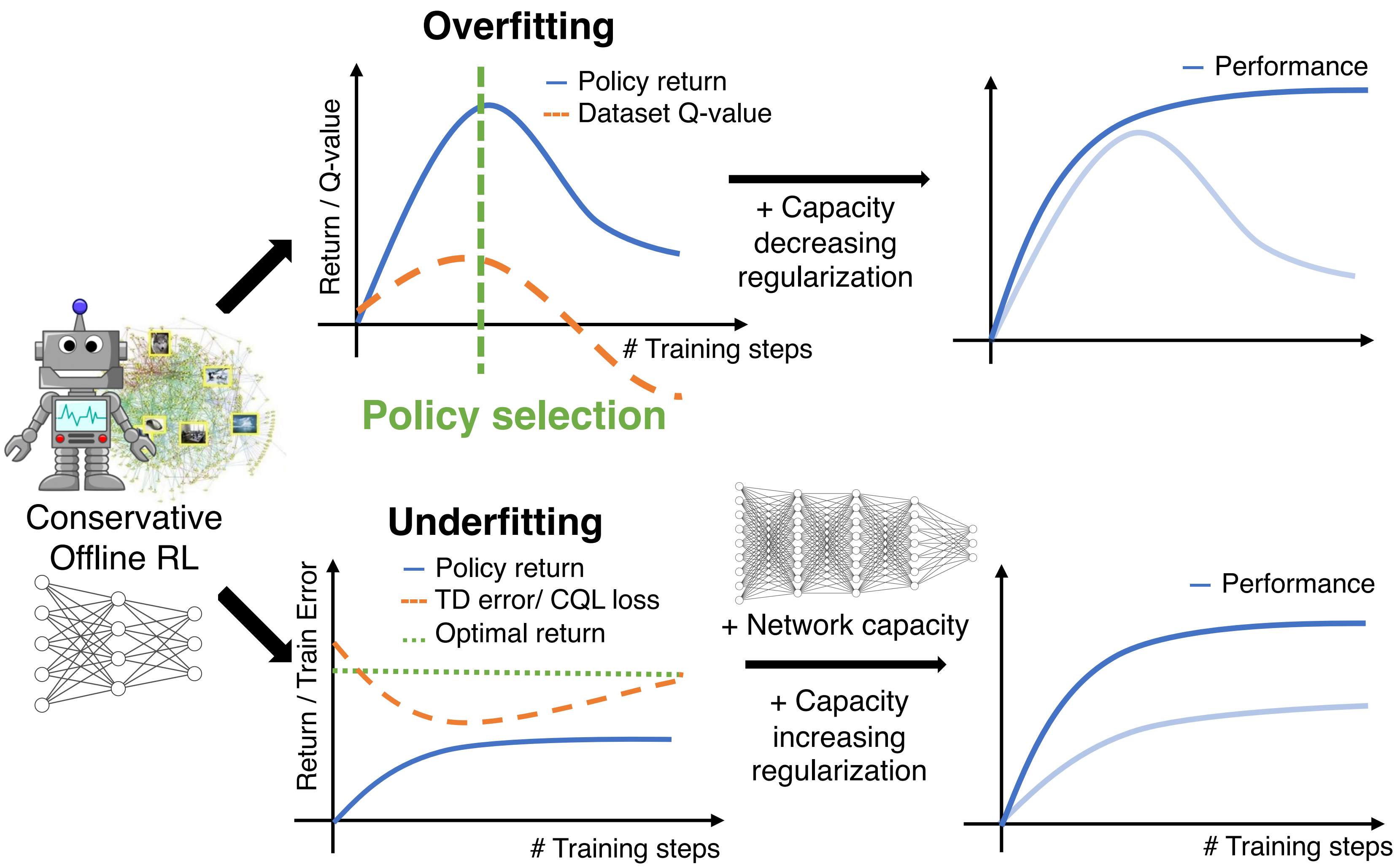
Let’s say we figure out the above, even then..

- ❑ Increased network capacity doesn’t mean better Q-functions
- ❑ How should we prevent overfitting?

We show how we can derive a workflow for a sub-class of offline RL algorithms.

K.*, Singh*, Tian*, Finn, Levine. **A Workflow for Offline Model-Free Robotic Reinforcement Learning.** In preparation, 2021.

A Workflow for Conservative Offline RL



- Track Q-values on the dataset to measure overfitting
- **(Early stopping)** Stop training Q-values start decreasing
- Large loss values to detect underfitting

K.*, Singh*, Tian*, Finn, Levine. **A Workflow for Offline Model-Free Robotic Reinforcement Learning.** In preparation, 2021.

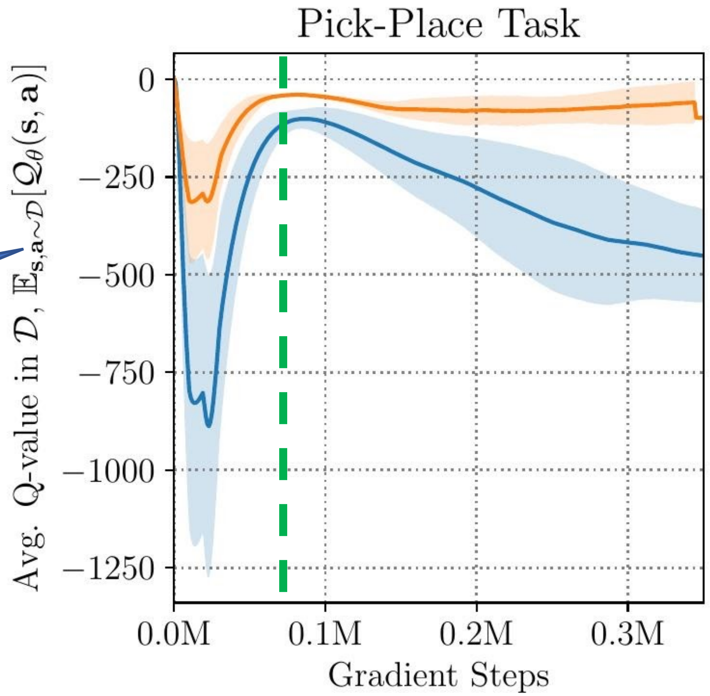
Overfitting in Conservative Offline RL

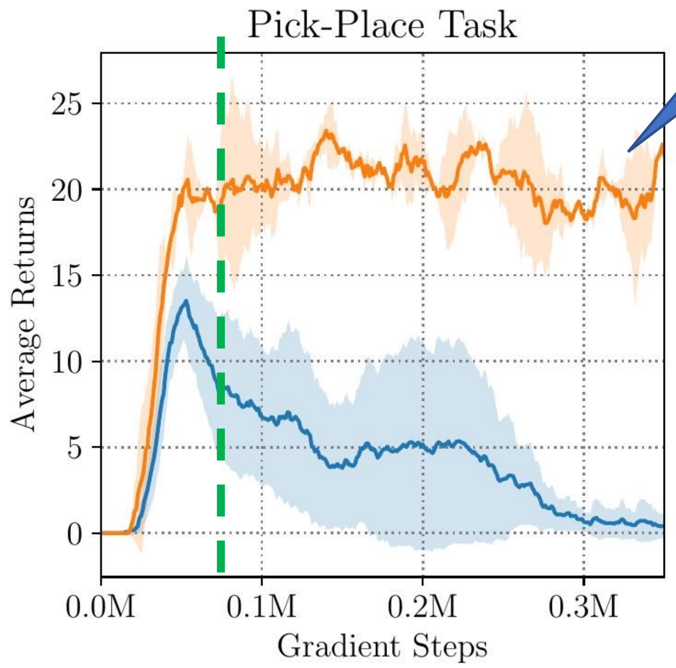
Overfitting

When Q-values start to decrease over training; stop training.
To address overfitting, we can use regularization (e.g., dropout, variational information bottleneck on features of the learned network)

$$\min_{\theta} \; \mathcal{L}_{\mathrm{CQL}}(\theta)+\beta\mathbb{E}_{\mathbf{s}\sim\mathcal{D}}\left[\mathrm{D}_{\mathrm{KL}}\left(\mathcal{N}(\phi_m(\mathbf{s}), \mathbf{diag}(\phi_\Sigma(\mathbf{s}))) \,||\, \mathcal{N}(0,\mathbb{I})\right)\right]$$

Q-values decrease over training





With VIB

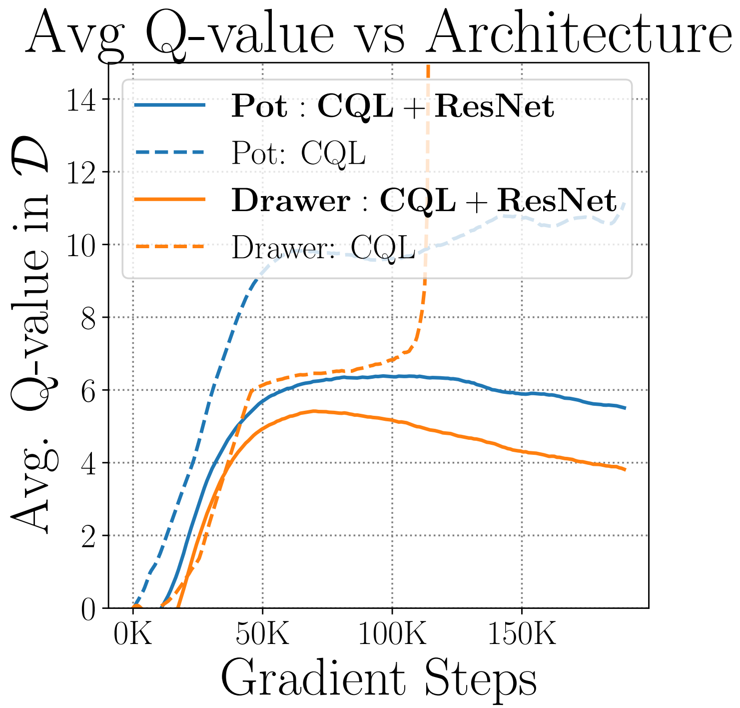
K.*, Singh*, Tian*, Finn, Levine. **A Workflow for Offline Model-Free Robotic Reinforcement Learning.** In preparation, 2021.

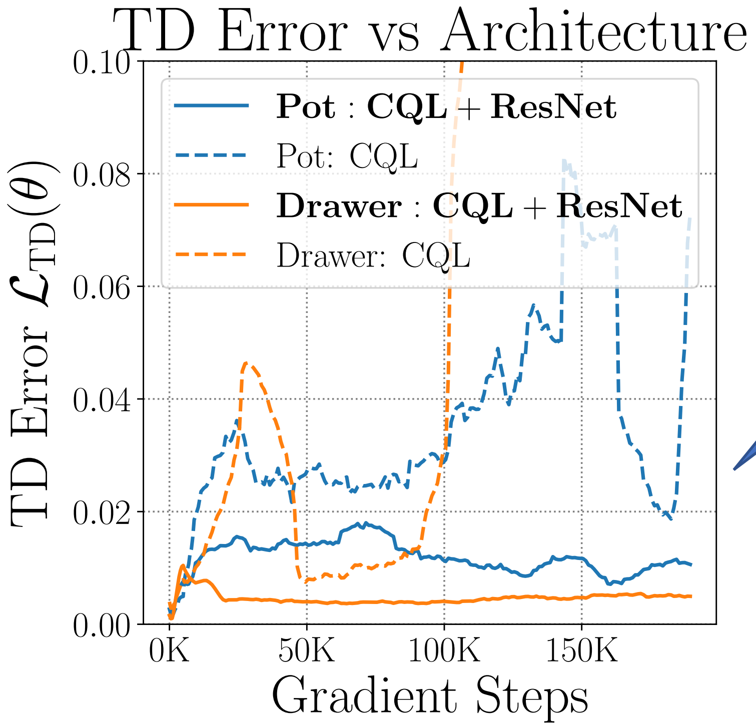
Underfitting in Conservative Offline RL

Underfitting

When training losses are high. In this case, we can use:
1. More expressive networks for the policy architecture
2. DR3 penalty for the critic (and maybe more expressive architectures)

Q-values increase over training





Large TD error with base net, lower with Resnet + DR3

K.*, Singh*, Tian*, Finn, Levine. **A Workflow for Offline Model-Free Robotic Reinforcement Learning.** In preparation, 2021.

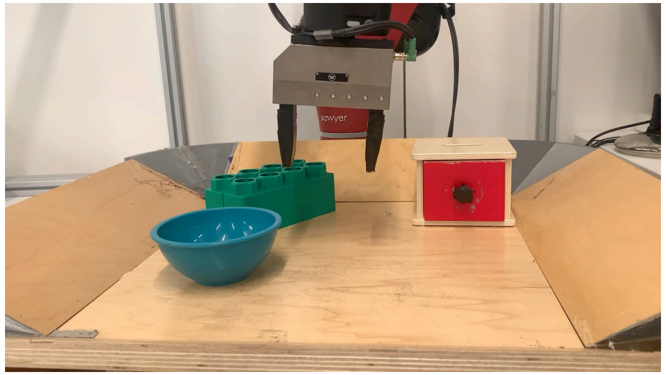
Tuning Underfitting on Real Robots

Scenario: Sawyer Manipulation tasks
(Place lid on pot, Open Drawer)

Tuning underfitting

0/12

9/12



Tuning underfitting

0/12

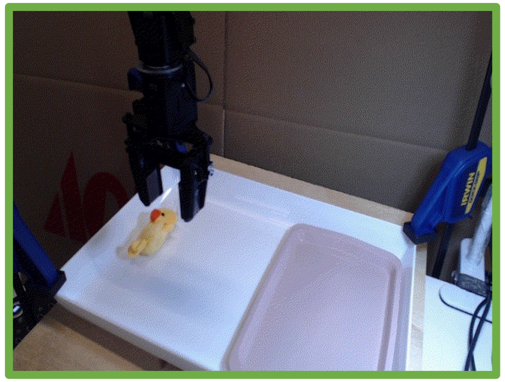
8/12

K.*, Singh*, Tian*, Finn, Levine. **A Workflow for Offline Model-Free Robotic Reinforcement Learning.** In preparation, 2021.

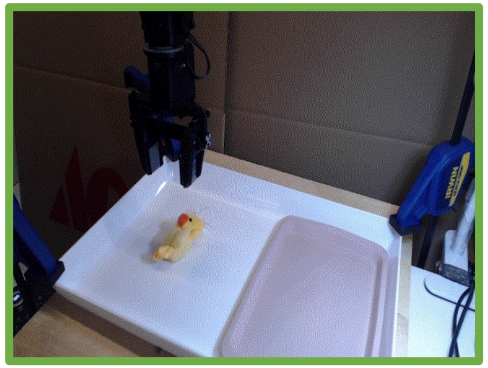
Tuning Overfitting on Real Robots

Scenario: Real WidowX pick & place

Baseline CQL **(3/9)**



Baseline CQL + Policy Selection **(7/9)**

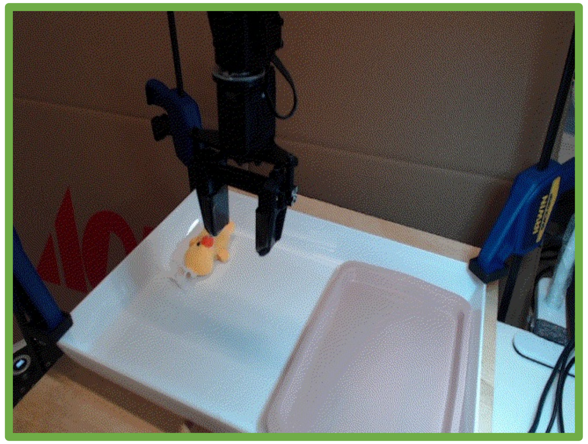
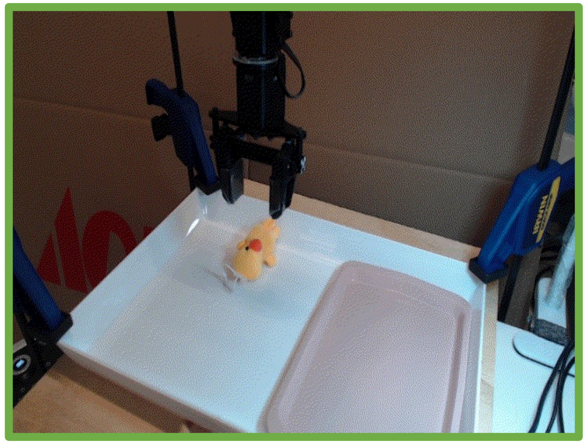
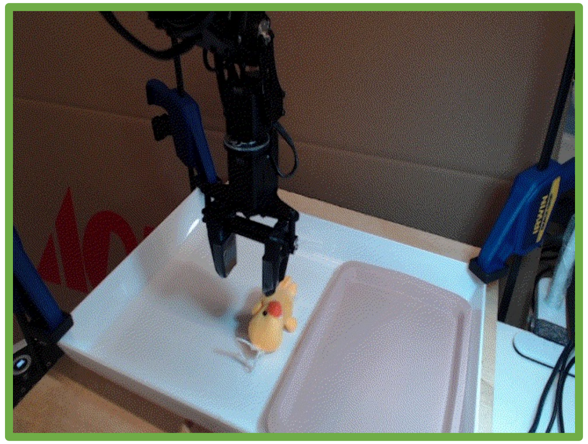


K.*, Singh*, Tian*, Finn, Levine. **A Workflow for Offline Model-Free Robotic Reinforcement Learning.** In preparation, 2021.

Tuning Overfitting on Real Robots

Scenario: Real WidowX pick & place

Baseline CQL + Overfitting Correction (VIB) + Early Stopping **(8/9)**


K.*, Singh*, Tian*, Finn, Levine. **A Workflow for Offline Model-Free Robotic Reinforcement Learning.** In preparation, 2021.

Summary and Conclusion

- Applying deep RL on real and new domains will (most likely) require making it's behavior understandable and amenable to easy tuning

- One way to do so is to understand how algorithms behave with neural networks:
  - Implicit Regularization of SGD, model class, etc. can hurt
  - Can add explicit regularization to tackle this problem.

- We should devise workflows (guidelines) for making it easy to use/tune deep RL.
  - We devise workflow for some algorithms and find it to work well on new, previously untuned problems.

Thank You!

Contact me at: aviralk@berkeley.edu

Work done with **Sergey Levine** (UC Berkeley), **George Tucker** (Google), **Rishabh Agarwal** (Google), **Dibya Ghosh** (UC Berkeley), **Anikait Singh** (UC Berkeley), **Stephen Tian** (UC Berkeley), **Chelsea Finn** (Stanford), **Tengyu Ma** (Stanford), **Aaron Courville** (MILA)