Learning Dynamics and Generalization in Deep Reinforcement Learning

Clare Lyle[1], Mark Rowland[2], Will Dabney[2], Marta Kwiatkowska[1], Yarin Gal[2]

[1]University of Oxford [2]DeepMind

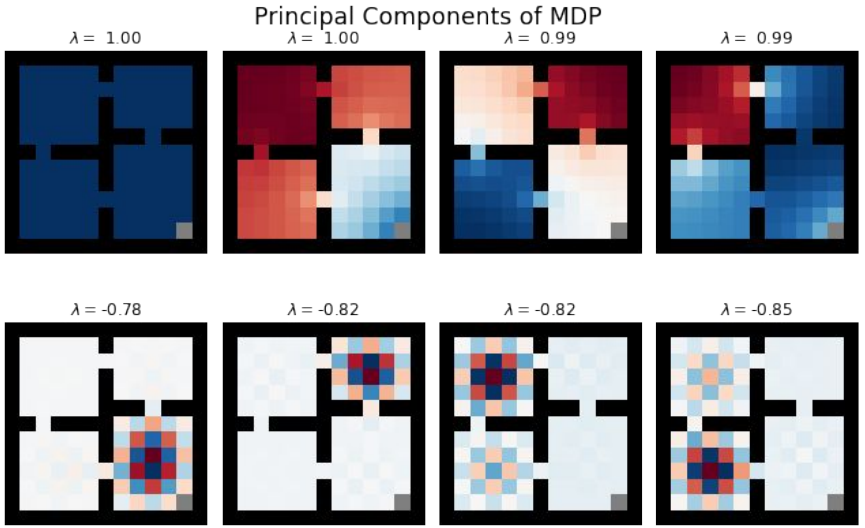
TD learning fits **non-smooth** components of the value function **first**.

Smoothness and eigendecompositions of P^{π}

Recall: eigendecomposition of P^{π} gives notion of **smoothness w.r.t. transition dynamics**.

Smooth functions ≈ eigenvectors corresponding to largest (**positive**) eigenvalues of P^{π}

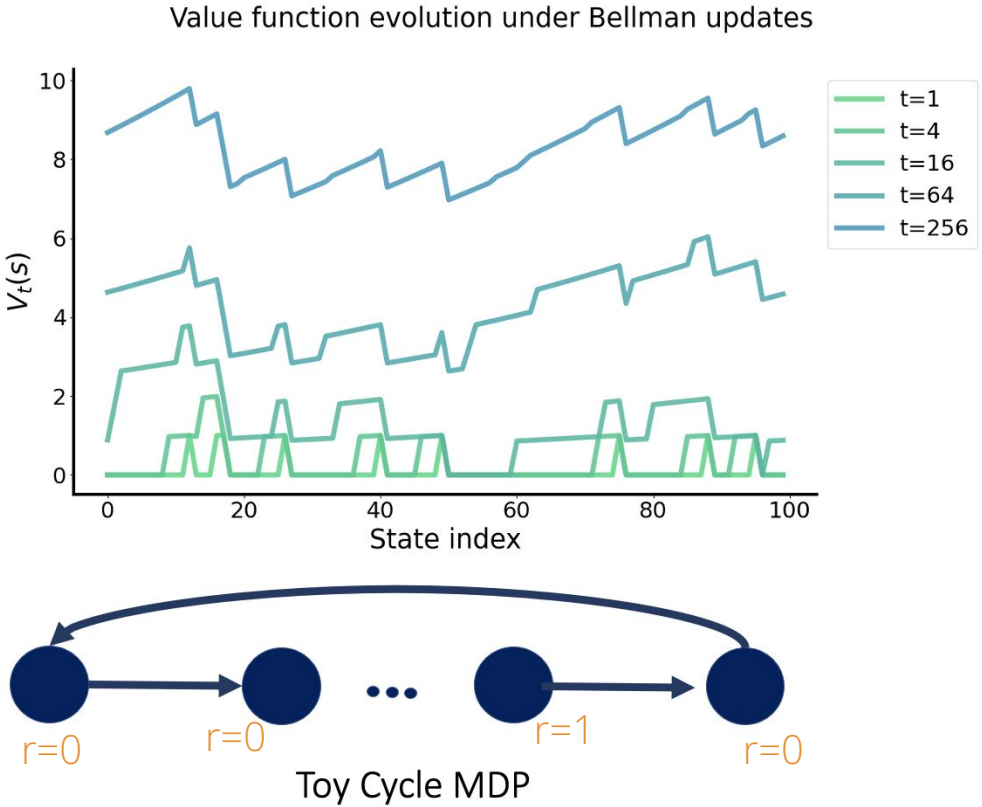
Non-smooth functions ≈ **negative** eigenvalues.



Dynamics of TD learning

TD methods learn about **nearby rewards first**, only pick up “global” structure of the value function later.

Rewards usually **less smooth** than the true value function.



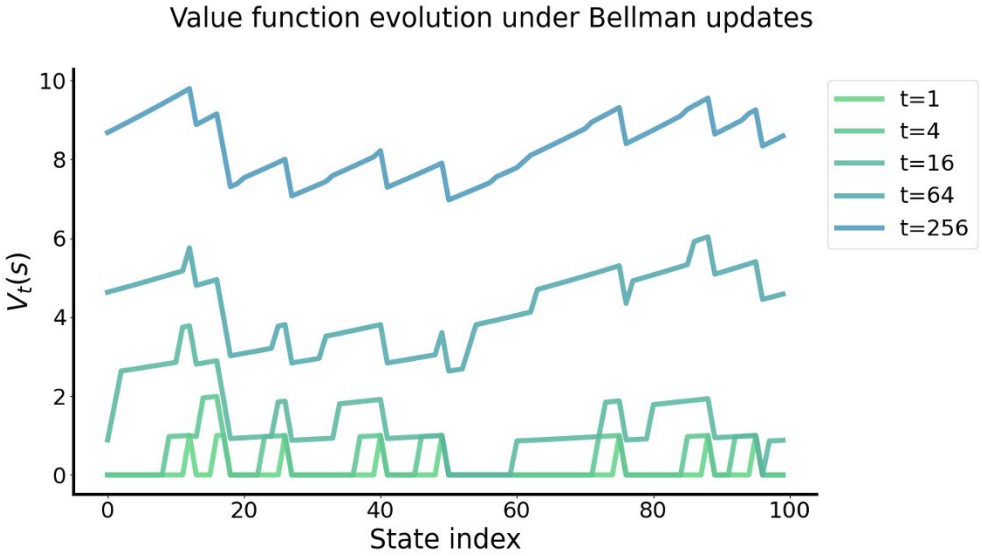
Dynamics of TD learning

Continuous-time approximation of value function evolution :

$$\partial_t V_t = -(I - \gamma P^\pi) V_t + R^\pi$$
$$V_t = \exp(-t(I - \gamma P^\pi))(V_0 - V^\pi) + V^\pi$$

Implies that value function error depends on eigendecomposition of P^π : V_t fits smooth components last.

$$\alpha_i^t - \alpha_i^\pi = \exp(-t(1 - \gamma\lambda_i))(\alpha_i^0 - \alpha_i^\pi)$$

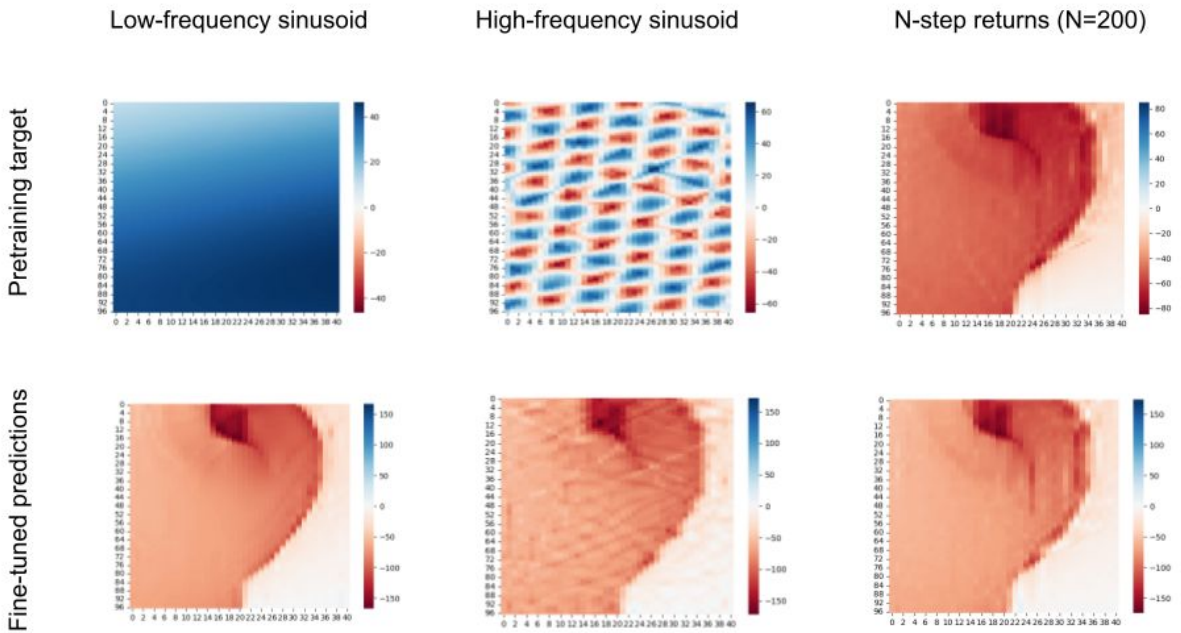


Early prediction targets determine **inductive biases** of deep neural networks.

Critical periods

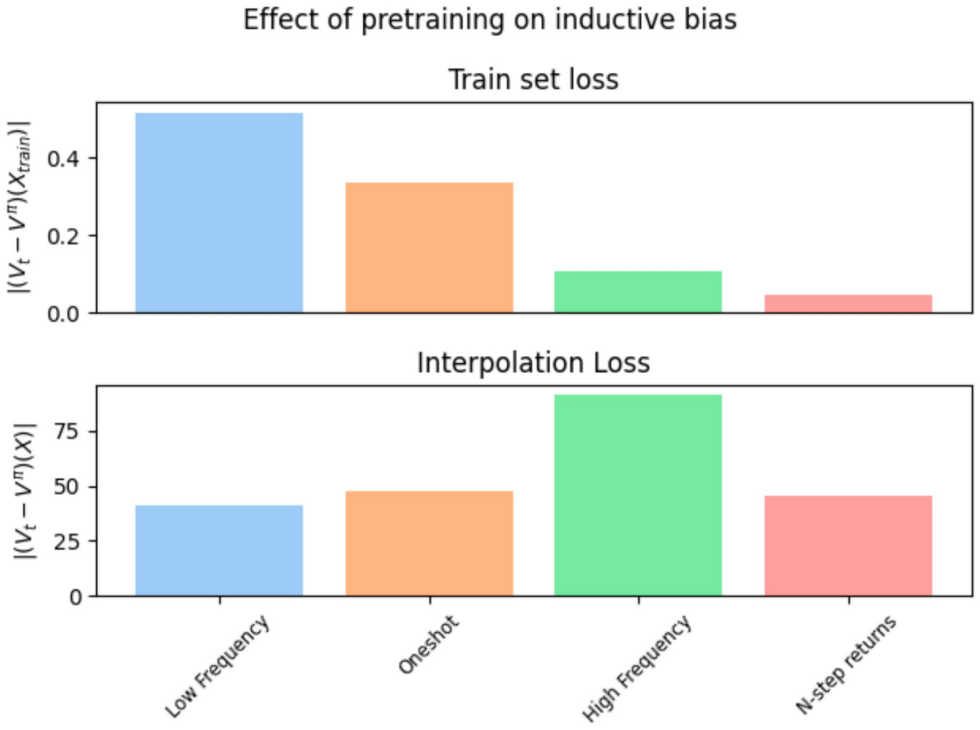
The **early learning period is crucial** for neural networks to incorporate **good inductive biases for generalization**.

Fitting a non-smooth function early in training can result in a **memorization bias**.



Early targets influence generalization error

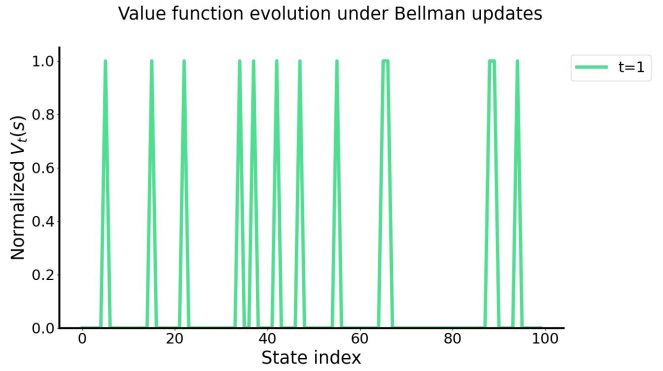
Networks **pre-trained** to fit **high-frequency** target functions go on to obtain **greater generalization error** when fine-tuned on a value function and evaluated on an interpolation task than those trained from scratch on the value function.



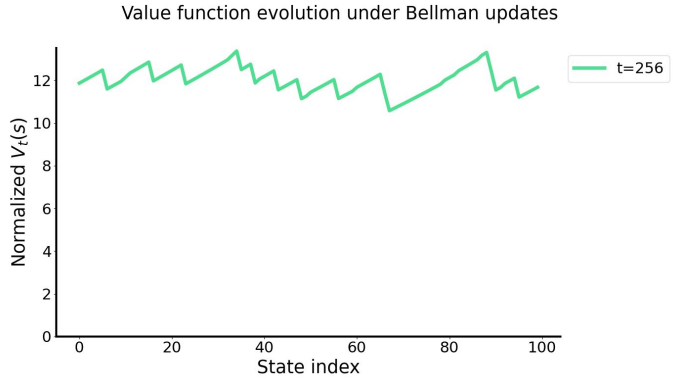
Fitting **non-smooth TD targets** early in training **hurts generalization** in value-based deep RL agents.

TD targets and deep RL

Temporal difference learning is hard for neural networks because they initially have to fit functions like this

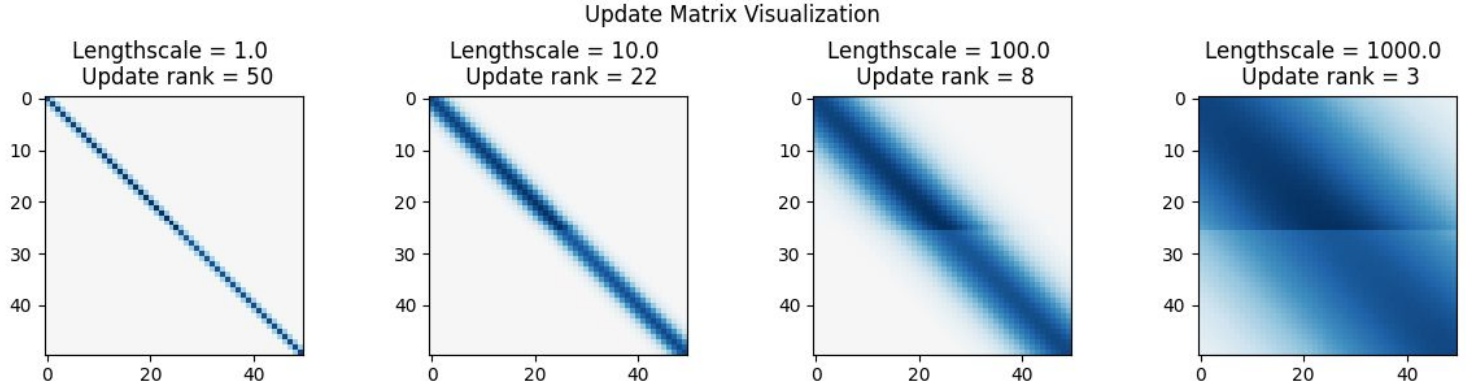


… and later on have to adapt to fit functions like this



Measuring memorization

Interference: the extent to which an **update** on **one state influences** the network's output on **other states.** If updating output for a state *x* does not change predictions for other states, then no generalization.

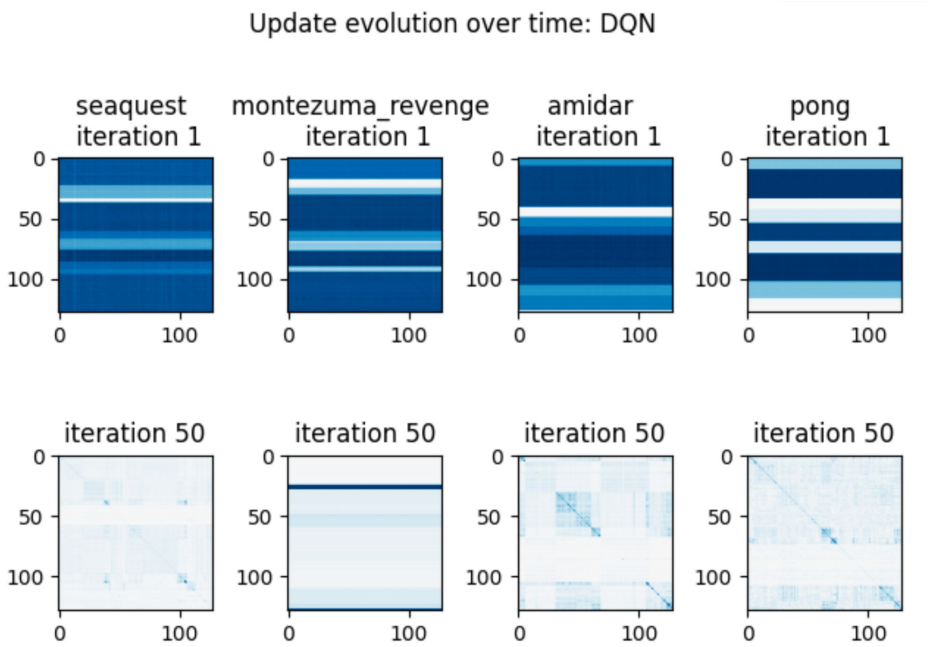


Update matrices: row i measures change in value output over all states after updating network to minimize loss at state x_i. Diagonal matrix implies no generalization; rank one matrix implies trivial generalization between inputs.

Memorization in value-based deep RL

Deep RL agents exhibit **decreasing interference** over the course of training**.**

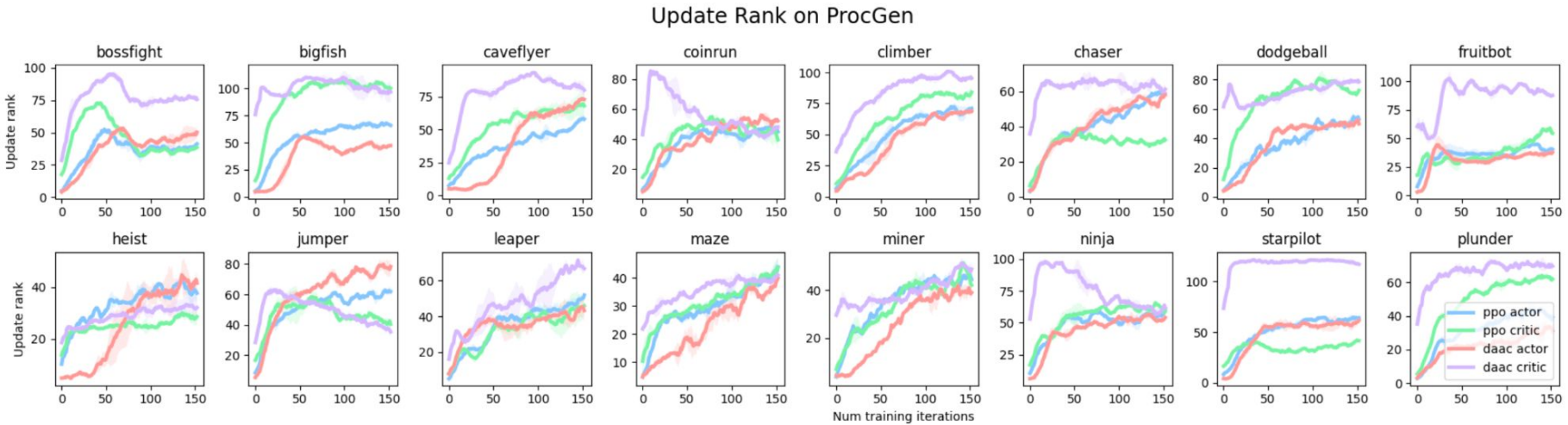
Suggests that networks are learning to memorize value function.



Update matrices: row i measures change in value output over all states after updating network to minimize loss at state x_i. Diagonal matrix implies no generalization; rank one matrix implies trivial generalization between inputs.

Value- vs policy-based RL

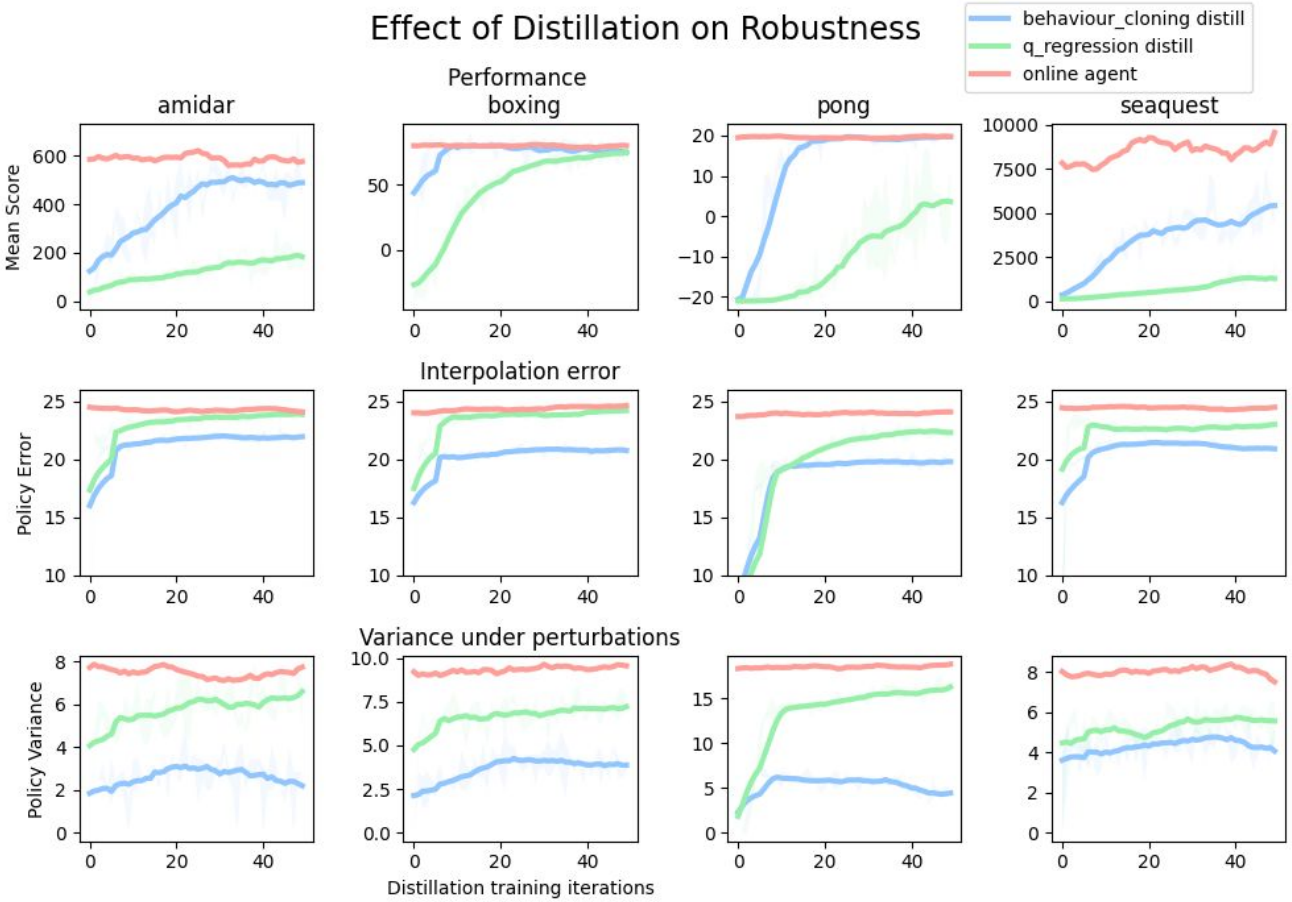
Value-based RL leads to **weaker interference** than **policy gradient** methods.



Distillation using a freshly initialized network **improves generalization**

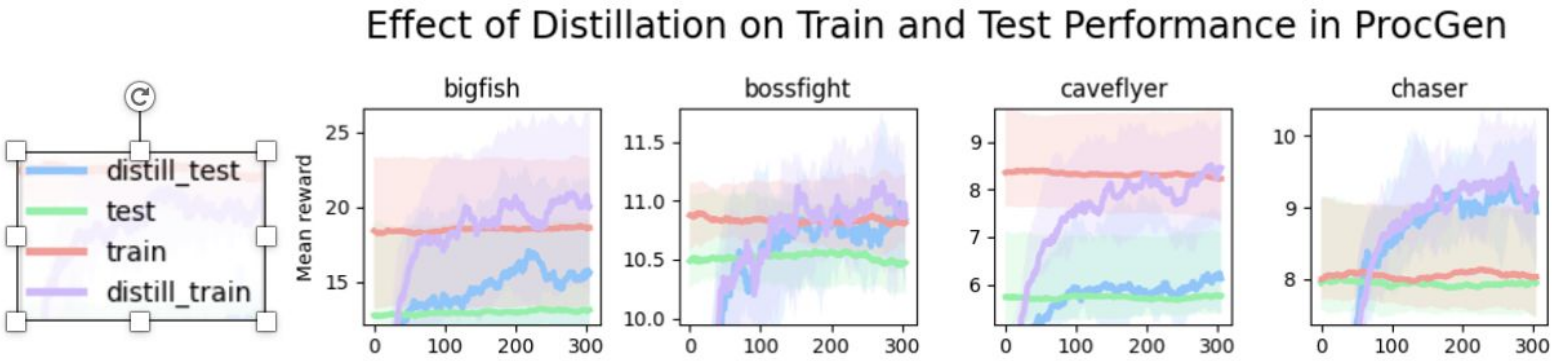
Distillation

Distillation improves robustness to perturbation and increases consistency on interpolations of the training data.



Policy Distillation

Policy distillation of a “fresh” network to mimic the behaviour of an actor-critic agent **improves generalization** to **new environments.**



Thanks!