On the Existence of Double-Descent in Reinforcement Learning

Richard Harnisch

27 January 2026

Outline

What is Double Descent?

- **Double Descent (DD)** is a phenomenon of models recovering generalization capacity after surpassing the interpolation threshold (Nakkiran et al. 2019).
- Classically, increasing model capacity leads to a U-shaped risk curve.
- However, modern machine learning models often operate in regimes where they can perfectly fit (interpolate) the training data.
- This can occur in both model size and training time regimes as well as multiple problem types in supervised learning (Belkin et al. 2019; Nakkiran et al. 2019).
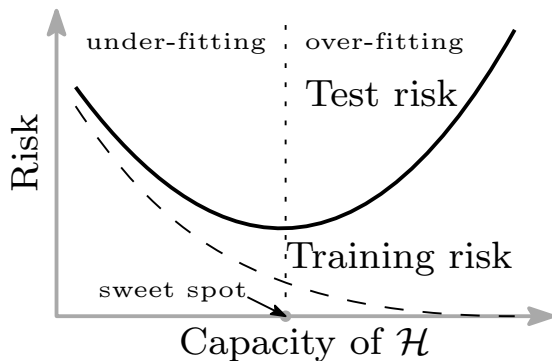
Classical U-shaped Risk Curve

Figure: Classical U-shaped risk curve arising from the bias-variance trade-off. Belkin et al. (2019).
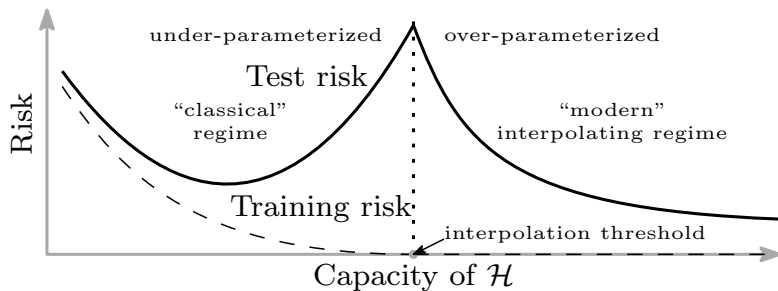
Double Descent Risk Curve

Figure: Double Descent risk curve exhibiting a second descent. Belkin et al. (2019).

DD in Reinforcement Learning

▶ Double Descent has not been studied much in Reinforcement Learning (RL) settings.

▶ There is some research on theoretical aspects of overparameterization in RL with regard to visited states (Brellmann et al. 2024).

▶ There is also an approach of using information theoretic metrics to analyze generalization in RL (Veselý, Todorov, and Sabatelli 2025).

Methods

- We are trying to emulate the supervised learning setup in RL: Train/test split, and plotting a "risk" curve.

- To mimick train and test sets, we use a family of seeded randomly generated maps, training on a subset of these maps and testing on a disjoint subset.
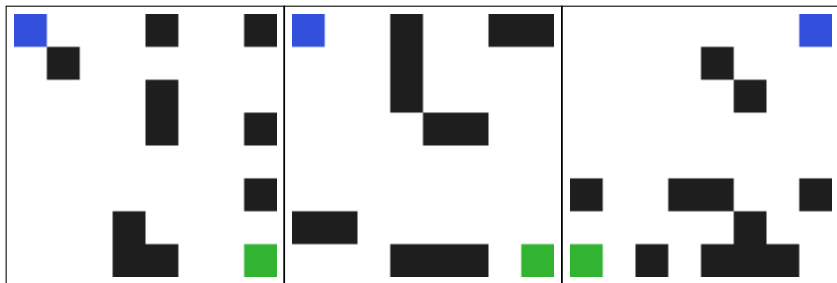

Figure: Examples of randomly generated training and validation environments, with agent (blue) and goal (green) tiles.

Metrics

- To plot risk curves, we need something to plot on the y-axis since we don't have a loss function in RL.

- We use the **average return** over the test maps as our metric.

- Further, we compute the trace of the **Fisher Information Matrix (FIM)** of the policy network.

- The FIM is defined as:

$$F = \mathbb{E}_{a,s \sim \pi_\theta} \left[\nabla_\theta \log \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)^T \right]$$

  where s, a are sampled from trajectories under the policy π_θ, collected on the training maps.

- Thus the FIM is the expected outer product of the score function (the gradient of the log-likelihood).

- The trace of the FIM gives a measure of the sensitivity of the policy to parameter changes, which can be related to generalization.

Agents & Training Setup

▶ We train two agent types: A DQN (Mnih et al. 2013) agent and a TRPO agent (Schulman et al. 2017), both using feedforward neural networks.

▶ DQNs implement Q-learning with experience replay using a neural network to estimate Q-values.

▶ TRPO is a policy gradient method that optimizes the policy directly while constraining updates using a trust region defined by the KL divergence.

▶ We vary the model capacity by changing the depth and width of the networks.

▶ Training is performed on a fixed set of training maps, with evaluation held-out test maps.

▶ We record the average return and trace of the FIM at various points during training to analyze the presence of Double Descent.
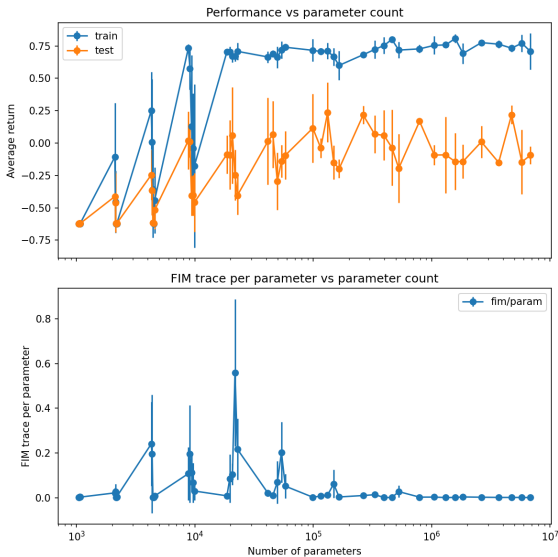
Results in Parametric Regime

Figure: Results from TRPO agents trained on 100 maps for 50k episodes.
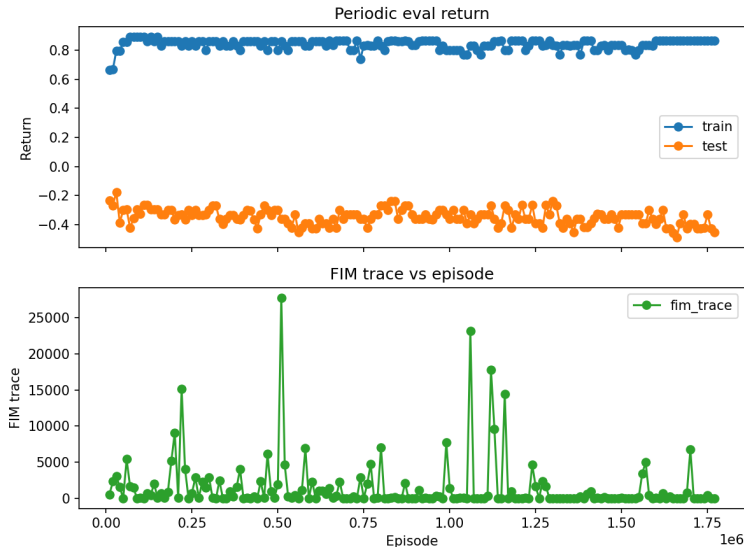
Results in Episodic Regime

Figure: Results of a TRPO agent trained on 100 maps for 1.8M episodes.

Discussion

- ▶ Unfortunately, we are unable to observe a clear Double Descent phenomenon in our experiments.
- ▶ There may be several reasons for this, including the complexity of reinforcement learning environments and the computational constraints limiting the scale of our experiments.
- ▶ We observe a spike in the FIM trace around the interpolation threshold. This indicates that the model is unstable in the parameter space, but stabilizes later.
- ▶ However, this does not translate to a clear second ascent in average return.

Limitations & Future Work

▶ We are constrained by limited access to compute, with training runs taking multiple days and wait times for available GPUs.

▶ One avenue for further work is exploring the transition between lacking generalization and perfect generalization.

▶ We observe no emerging generalization gap when training on 1,000 maps.

▶ Further, training on larger maps as well as different algorithms could be explored, as well as stochastic transitions.

References I

📄 Belkin, Mikhail et al. (July 2019). "Reconciling modern machine-learning practice and the classical bias–variance trade-off". In: *Proceedings of the National Academy of Sciences* 116.32, pp. 15849–15854. ISSN: 1091-6490. DOI: 10.1073/pnas.1903070116. URL: http://dx.doi.org/10.1073/pnas.1903070116.

📄 Brellmann, David et al. (2024). "On Double Descent in Reinforcement Learning with LSTD and Random Features". In: arXiv: 2310.05518 [cs.LG]. URL: https://arxiv.org/abs/2310.05518.

📄 Mnih, Volodymyr et al. (2013). "Playing Atari with Deep Reinforcement Learning". In: arXiv: 1312.5602 [cs.LG]. URL: https://arxiv.org/abs/1312.5602.

📄 Nakkiran, Preetum et al. (2019). *Deep Double Descent: Where Bigger Models and More Data Hurt*. arXiv: 1912.02292 [cs.LG]. URL: https://arxiv.org/abs/1912.02292.

References II

📄 Schulman, John et al. (2017). "Trust Region Policy Optimization". In: arXiv: 1502.05477 [cs.LG]. URL: https://arxiv.org/abs/1502.05477.

📄 Veselý, Viktor, Aleksandar Todorov, and Matthia Sabatelli (2025). *On The Presence of Double-Descent in Deep Reinforcement Learning*. arXiv: 2511.06895 [cs.LG]. URL: https://arxiv.org/abs/2511.06895.

Thank You!

Questions?

Environment Details

- Each map is 8×8 tiles, with the agent in the top left and the goal in the bottom right unless randomized.
- Every other tile has a 20% chance of being blocked, with a guaranteed path.
- At each timestep, the agent receives a reward given by:

$$r_t = \begin{cases} 1, & \text{if goal reached at time } t \\ \Phi(s_{t+1}) - \Phi(s_t) - 0.01, & \text{otherwise} \end{cases}$$

  where $\Phi(s) = \frac{\text{Euclidean distance to goal}}{100}$ is the potential function.
- Episodes are truncated at 64 timesteps.
- Action space: up, right, down, left, deterministic transitions.
- Observation space: 8×8 grid with one-hot encoded tiles, frame stacking of 2 frames ($8 \times 8 \times 4 \times 2 = 512$ bits).

Training Configurations

- On the following configurations, we trained agents of varying widths between 3 and 8 and depths between 4 and 1024:

Type	Train Maps	Training Episodes	Start & Goal Position
TRPO	10	10,000	Standard[1]
DQN	50	20,000	Standard
DQN	100	50,000	Standard
TRPO	50	20,000	Randomized
TRPO	100	50,000	Randomized
TRPO	500	1,000,000	Standard

- On the following configurations, we trained agents of depth 3 and widths 4, 16, 64, 256, and 1024 for unlimited time:

Type	Train Maps	Start & Goal Position
TRPO	50	Standard
TRPO	100	Standard

---

[1]Agent starts in top left, goal in bottom right

The Fisher Information Matrix

▶ The **Fisher Information Matrix** (FIM) is defined as:

$$F(\theta) = \mathbb{E}_{a,s \sim \pi_\theta} \left[\nabla_\theta \log \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)^T \right]$$

▶ s, a are sampled under the stochastic policy π_θ defined by parameters θ.

▶ $\nabla_\theta \log \pi_\theta(a|s)$ is the gradient of the log-probability of the action given the state with respect to the policy parameters

▶ A FIM where $\theta = [\theta_1, \theta_2]^T$ is therefore a 2×2 matrix given by

$$\mathbb{E} \begin{bmatrix} \left(\frac{\partial}{\partial \theta_1} \log \pi_\theta(a|s) \right)^2 & \frac{\partial}{\partial \theta_1} \log \pi_\theta(a|s) \frac{\partial}{\partial \theta_2} \log \pi_\theta(a|s) \\ \frac{\partial}{\partial \theta_2} \log \pi_\theta(a|s) \frac{\partial}{\partial \theta_1} \log \pi_\theta(a|s) & \left(\frac{\partial}{\partial \theta_2} \log \pi_\theta(a|s) \right)^2 \end{bmatrix}$$

Computing the FIM's Trace

▶ We do not need to form the full $|\theta| \times |\theta|$ matrix!

▶ Recall $F = \mathbb{E}_{a,s \sim \pi_\theta} \left[\nabla_\theta \log \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)^T \right]$, and let $g = \nabla_\theta \log \pi_\theta(a|s)$. Then $F = \mathbb{E}[gg^T]$.

▶ Since the trace of a matrix is the sum of its diagonal entries $\sum_i F_{ii}$, we have

$$\text{tr}(F) = \sum_i \mathbb{E}[g_i^2] = \mathbb{E}\left[\sum_i g_i^2 \right] = \mathbb{E}[g^T g] = \mathbb{E}[\|g\|_2^2].$$

▶ We can thus form a Monte Carlo estimate of the trace of the FIM by sampling states from collected trajectories and actions from the policy in that state:

$$\text{tr}(F) \approx \frac{1}{N} \sum_{i=1}^{N} \|\nabla_\theta \log \pi_\theta(a_i|s_i)\|_2^2$$

Algorithms: DQN and TRPO

- **Deep Q-Networks (DQN)** (Mnih et al. 2013) are value-based and approximate the optimal action-value function $Q^*(s, a)$ using a neural network. The agent selects actions based on an ϵ-greedy policy derived from the Q-values.

- **Trust Region Policy Optimization (TRPO)** (Schulman et al. 2017) is a policy gradient method that optimizes the policy directly.

- It uses a surrogate objective function given by the expected advantage $\mathbb{E}_{s,a \sim \pi_{\theta_{\text{old}}}} \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} A^{\pi_{\theta_{\text{old}}}}(s, a) \right]$.

- TRPO constrains the policy updates using a trust region defined by the Kullback-Leibler (KL) divergence to ensure stable learning by $D_{KL}(\pi_{\theta_{\text{old}}} || \pi_\theta) \leq \delta$ for some small δ.