Generalization in Deep Reinforcement Learning

Presenter: Jake Grigsby

University of Virginia
https://qdata.github.io/deep2Read/

202008

Generalization in Supervised Learning

Generalization in supervised learning (SL) is measured by performance on a held-out test set sampled from the same distribution as your training data.

Generalization error, E_G = test error $-$ train error

Given:

- $(\mathcal{X}_{train}, \mathcal{Y}_{train}), (\mathcal{X}_{test}, \mathcal{Y}_{test}) \sim \mathcal{D}$
- Model f, loss function $l(f(x), y)$

$$E_G(f) = \frac{1}{|\mathcal{X}_{test}|} \sum_{i=1}^{|\mathcal{X}_{test}|} l(f(x_{test,i}), y_{test,i}) - \frac{1}{|\mathcal{X}_{train}|} \sum_{j=1}^{|\mathcal{X}_{train}|} l(f(x_{train,j}), y_{train,j})$$

What Does Generalization Look like in RL?

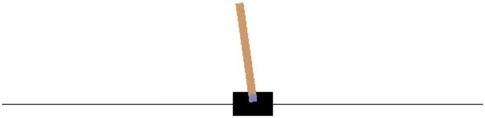
We measure generalization in SL to be confident our models will perform well on inputs they did not see during training.

What is the equivalent goal in RL?

- Performance on unseen states?
  - Yes, but this is only part of the picture
- Robust to slight variations in environment conditions
  - Changes to goals, dynamics, starting positions and observations.

Changing Environmental Conditions

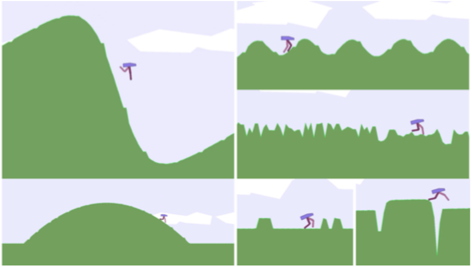
As an example, let's think about the classic cartpole task



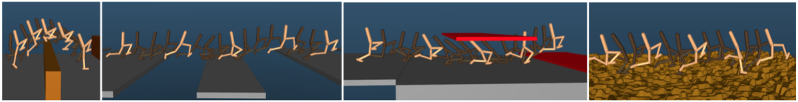
How will a policy successfully trained to balance the pole adjust to:

- Changes in the height or mass of the pole?
- Changes in the speed of the cart (friction)?
- Changes in the color of the cart or background (pixel-based)?

Changing Environmental Conditions



Will the walker agent trained to traverse the terrain on the left be able to walk down the stairs on the right?



Will the hopper agent trained to navigate one obstacle course be able to complete the others?

What Does Generalization Look like in RL?

- If the agent truly understood the task it was solving, these changes would not be a problem.
- Instead, if the agent has memorized a policy that worked during training, any changes to the environment could be a real challenge.

We want to learn policies that avoid overfitting to the specifics of the environment they were trained in.

This is "zero-shot generalization"

Partially Observed MDP

Definition

A Partially Observed Markov Decision Process (POMDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, R, \rho_0, \Omega, O, \gamma)$, consists of:

- $\mathcal{S}$, a set of states
- $\mathcal{A}$, a set of actions
- a dynamics function $T : \mathcal{S} x \mathcal{A} x \mathcal{S} \rightarrow [0, 1]$
- a reward function $R : \mathcal{S} x \mathcal{A} \rightarrow \mathbb{R}$
- an initial state distribution, ρ_0
- Ω, a set of observations, which in the partially observed case $\neq \mathcal{S}$
- an observation function $O : \mathcal{S} x \Omega \rightarrow [0, 1]$
- the discount factor $\gamma \in [0, 1)$

RL Objective

The goal of Reinforcement Learning is to find a policy $\pi(a|s)$ that maximizes the return:

$$\pi^* = \underset{\pi}{argmax} \underset{\tau \sim \pi}{\mathbb{E}} \left[\sum_{t=0}^{t=\infty} \gamma^t R_t \right]$$

Where τ is a trajectory of experience (the sequence of states and actions the agent experiences)

The return of a policy π in a POMDP $\mathcal{M}$ is denoted $\eta_{\mathcal{M}}(\pi)$

Quantifying Generalization in RL

More formally, these slight changes in the environment we've been talking about create a family of POMDPS to solve:

$$D = \{\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, ...\}$$

We can define generalization in RL as the gap in performance between a training and test set of POMDPS:

$$\mathcal{M}_{train}, \mathcal{M}_{test} \sim \mathcal{D}$$

$$E_G(\pi) = \frac{1}{|\mathcal{M}_{test}|} \sum_{i=1}^{|\mathcal{M}_{test}|} \eta_{\mathcal{M}_{test},i}(\pi) - \frac{1}{|\mathcal{M}_{train}|} \sum_{j=1}^{|\mathcal{M}_{train}|} \eta_{\mathcal{M}_{train},j}(\pi)$$

Improving Generalization in RL

There are three main approaches to improving generalization in RL:

1. Data Augmentation
2. Domain Randomization
3. Procedural Generation

Observational Overfitting

In a POMDP, we have a set of states ($\mathcal{S}$) and a set of observations (Ω)

- States are a clear representation of all the relevant information needed to make accurate decisions
- Observations are what the agent actually gets to see. Information is often hidden because of limited memory, inaccurate sensors, or useless noise.

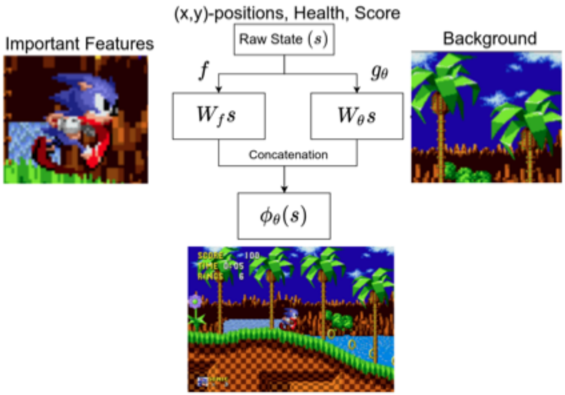
Let a function $\phi : \mathcal{S} \rightarrow \Omega$ define how the environment emits observations for a given state. We can break ϕ up into three subcomponents [10]:

$$\phi(s) = h(f(s), g(s))$$

f maps the important state information to the dimension of the observation, g outputs unimportant/ungeneralizable information, and h combines them in some way to make the final observation.

Observational Overfitting

Let's look at an example:



(a)

Observational Overfitting

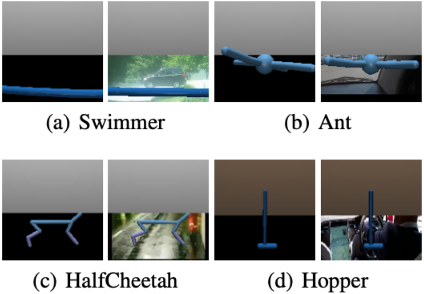
Observational overfitting occurs when the policy becomes overly dependent on features from $g(s)$.

Figure 1: Example of observational overfitting in Sonic. Saliency maps highlight (in red) the top-left timer and background objects because they are correlated with progress.

A similar problem occurs with ungeneralizable high-frequency features in SL [1] [3] [11]

Observational Overfitting

Is observational overfitting common in practice? We can test this by replacing useless details (like the background) with random noise and natural images ([7]). Performance is dramatically reduced!



(a) Swimmer (b) Ant

(c) HalfCheetah (d) Hopper

(a) Breakout (b) Gravitar

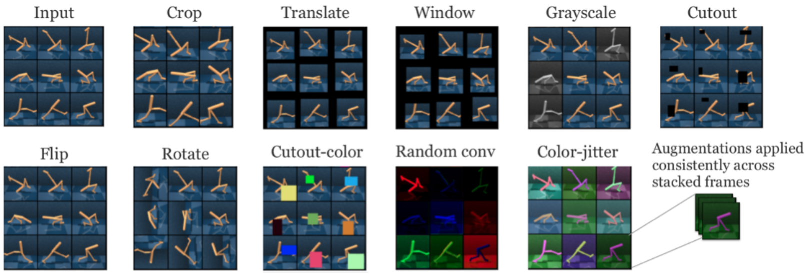
Figure 2: Atari frames, original (left), Gaussian noise (center), and with natural video embedded as background (right).

Data Augmentation

We can prevent observational overfitting by making it difficult to rely on features from g.

Data Augmentation [13] [15] [18] [12]:

- Convert gradient updates on observations into an expectation over a set of transformations of those observations
  - Make it hard to rely on background noise by consistently changing the way we present the observation to the agent



Data Augmentation

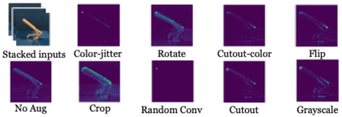
Data Augmentation ([13]), can match sample efficiency of model-based methods:

500K step scores	RAD	CURL	PlaNet	Dreamer	SAC+AE	SLACv1	Pixel SAC	State SAC
Finger, spin	**947** ± 101	926 ± 45	561 ± 284	796 ± 183	884 ± 128	673 ± 92	192 ± 166	923 ± 211
Cartpole, swing	**863** ± 9	845 ± 45	475 ± 71	762 ± 27	735 ± 63	-	419 ± 40	848 ± 15
Reacher, easy	**955** ± 71	929 ± 44	210 ± 44	793 ± 164	627 ± 58	-	145 ± 30	923 ± 24
Cheetah, run	**728** ± 71	518 ± 28	305 ± 131	570 ± 253	550 ± 34	640 ± 19	197 ± 15	795 ± 30
Walker, walk	**918** ± 16	902 ± 43	351 ± 58	897 ± 49	847 ± 48	842 ± 51	42 ± 12	948 ± 54
Cup, catch	**974** ± 12	959 ± 27	460 ± 380	879 ± 87	794 ± 58	852 ± 71	312 ± 63	974 ± 33
100K step scores								
Finger, spin	**856** ± 73	767 ± 56	136 ± 216	341 ± 70	740 ± 64	693 ± 141	224 ± 101	811 ± 46
Cartpole, swing	**828** ± 27	582 ± 146	297 ± 39	326 ± 27	311 ± 11	-	200 ± 72	835 ± 22
Reacher, easy	**826** ± 219	538 ± 233	20 ± 50	314 ± 155	274 ± 14	-	136 ± 15	746 ± 25
Cheetah, run	**447** ± 88	299 ± 48	138 ± 88	235 ± 137	267 ± 24	319 ± 56	130 ± 12	616 ± 18
Walker, walk	**504** ± 191	403 ± 24	224 ± 48	277 ± 12	394 ± 22	361 ± 73	127 ± 24	891 ± 82
Cup, catch	**840** ± 179	769 ± 43	0 ± 0	246 ± 174	391 ± 82	512 ± 110	97 ± 27	746 ± 91

Data Augmentation

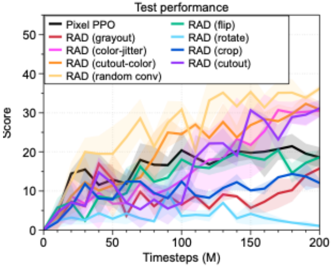
However, not all combinations of transformations are helpful:

	Crop	Gray scale	Rotate	Cutout	Color-jitter	Flip
Crop	920	849	635	855	797	650
Grayscale	856	175	349	187	214	231
Rotate	604	403	268	293	392	394
Cutout	722	206	376	215	31	284
Color-jitter	831	227	420	407	194	265
Flip	828	210	391	244	264	223



(a) Scores on DMControl500k for Walker, walk.

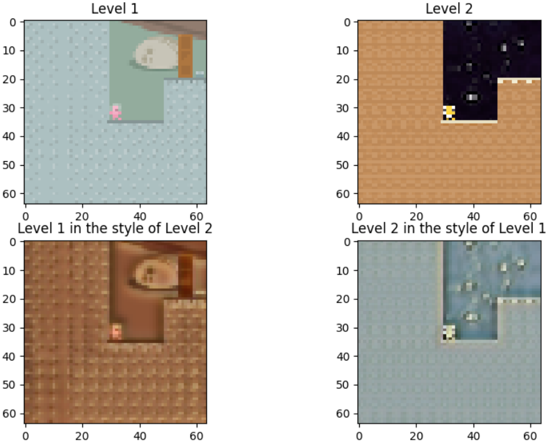
(b) Spatial attention map of augmentations for Walker, walk.



We need to pick transformations that preserve the quantity we are fitting ($Q(s,a)$ or $\pi(a|s)$) [15]

Data Augmentation

Another approach: reduce dependence on features from g, *while also improving performance on other tasks with different visual styles*:



Use arbitary style transfer [4] to map the g features from one task onto all the other tasks in our dataset.

Domain Randomization

Data Augmentation deals with generalization in observation space, but how do we improve generalization across core environment dynamics ($T(s, a, s')$)?

Convert a single task into a distribution of tasks by randomizing as many aspects of the environment as possible, and resampling those aspects every time we reset

This is called Domain Randomization [2] [14]

Domain Randomization

For some tasks, we cam implement this by simply reseeding the RNG after every environment reset [6]

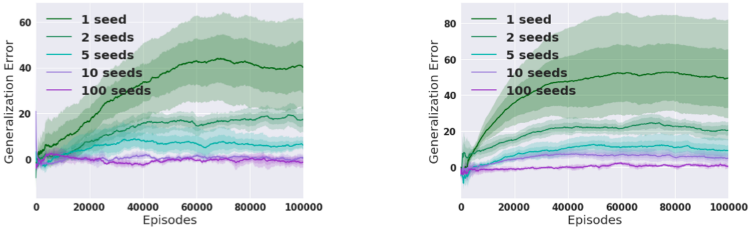
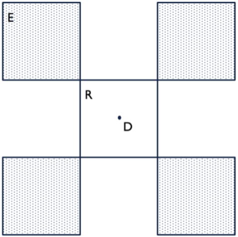


Figure 1: The 6-dim Acrobot (left) and Pixel Acrobot (right), varying number of train seeds from 1 to 100, with $\gamma = 0.99$. Averaged over 5 runs. Trained for 10K episodes (6-dim) and 100K episodes (Pixel).

Domain Randomization

Explicitly expand the parameter range of each component of the environment (wind, friction, agent mass, etc), and sample from that range after each reset [2] [14]. Multiple training types [5]:

1. **Deterministic** (D): every parameter is held fixed. When the environment is reset, only the state is reset.
2. **Random** (R): parameters are sampled (uniformly) after every reset from a reasonable range (in-distribution interpolation)
3. **Extreme** (E): parameters are sampled from a range twice as wide as the Random version (out-of-distribution extrapolation)



Domain Randomization

Table 2. Generalization performance (in % success) of each algorithm, averaged over all environments (mean and standard deviation over five runs).

Algorithm	Architecture	Default	Interpolation	Extrapolation
A2C	FF	78.14 ± 6.07	76.63 ± 1.48	63.72 ± 2.08
	RC	81.25 ± 3.48	72.22 ± 2.95	60.76 ± 2.80
PPO	FF	78.22 ± 1.53	70.57 ± 6.67	48.37 ± 3.21
	RC	26.51 ± 9.71	41.03 ± 6.59	21.59 ± 10.08
EPOpt-A2C	FF	2.46 ± 2.86	7.68 ± 0.61	2.35 ± 1.59
	RC	9.91 ± 1.12	20.89 ± 1.39	5.42 ± 0.24
EPOpt-PPO	FF	85.40 ± 8.05	85.15 ± 6.59	59.26 ± 5.81
	RC	5.51 ± 5.74	15.40 ± 3.86	9.99 ± 7.39
RL^2-A2C	RC	45.79 ± 6.67	46.32 ± 4.71	33.54 ± 4.64
RL^2-PPO	RC	22.22 ± 4.46	29.93 ± 8.97	21.36 ± 4.41

Domain Randomization

Domain randomization can even be enough to generalize from simulation to the real world [2]:

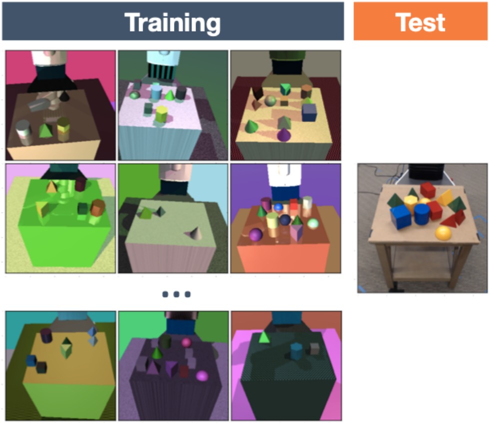


Fig. 1. Illustration of our approach. An object detector is trained on hundreds of thousands of low-fidelity rendered images with random camera positions, lighting conditions, object positions, and non-realistic textures. At test time, the same detector is used in the real world with no additional training.

Procedural Generation

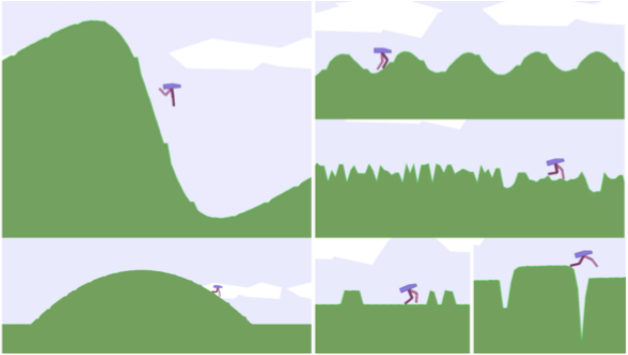
Leverage procedural generation to create as many training tasks as we need. This is becoming a staple of recent RL benchmarks:

- Procgen [9] [8]

Figure 1. Screenshots from each game in Procgen Benchmark.

Procedural Generation

- POET [16]
  - Locomotion across varying terrain
  - Also involves curriculum-learning/open-endedness



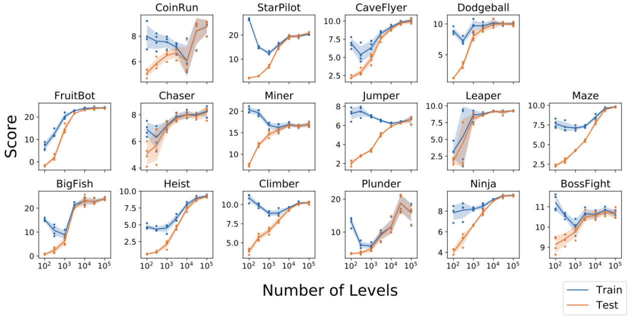
Procedural Generation

- ALLSTEPS [17]
  - Continuous control through increasingly difficult environments

Figure 1: *Virtual human (left), Cassie (middle), and Monster (right) walk across randomly generated stepping-stone terrain.*

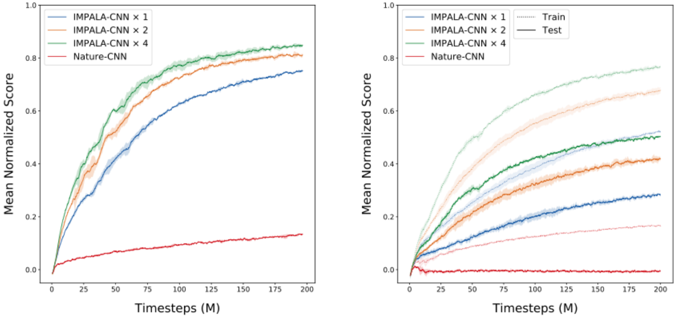
Procedural Generation

Procedural generation gives us an opportunity to measure the impact of training set size on generalization [8]:



Procedural Generation

Model architecture/capacity - often a forgotten implementation detail in model-free RL - starts to matter more when learning distributions of tasks [8]:



Next Time

Sample efficiency and offline learning, with model-based RL as a form of data augmentation.

References I

Jason Jo and Yoshua Bengio. "Measuring the tendency of CNNs to learn surface statistical regularities". In: *arXiv preprint arXiv:1711.11561* (2017).

Josh Tobin et al. "Domain randomization for transferring deep neural networks from simulation to the real world". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Sept. 2017). DOI: 10.1109/iros.2017.8202133. URL: http://dx.doi.org/10.1109/IROS.2017.8202133.

Robert Geirhos et al. "ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness". In: *arXiv preprint arXiv:1811.12231* (2018).

Xueting Li et al. *Learning Linear Transformations for Fast Arbitrary Style Transfer*. 2018. arXiv: 1808.04537 [cs.CV].

Charles Packer et al. "Assessing generalization in deep reinforcement learning". In: *arXiv preprint arXiv:1810.12282* (2018).

References II

- Amy Zhang, Nicolas Ballas, and Joelle Pineau. "A dissection of overfitting and generalization in continuous reinforcement learning". In: *arXiv preprint arXiv:1806.07937* (2018).
- Amy Zhang, Yuxin Wu, and Joelle Pineau. "Natural environment benchmarks for reinforcement learning". In: *arXiv preprint arXiv:1811.06032* (2018).
- Karl Cobbe et al. "Leveraging procedural generation to benchmark reinforcement learning". In: *arXiv preprint arXiv:1912.01588* (2019).
- Karl Cobbe et al. "Quantifying generalization in reinforcement learning". In: *International Conference on Machine Learning*. 2019, pp. 1282–1289.
- Xingyou Song et al. "Observational overfitting in reinforcement learning". In: *arXiv preprint arXiv:1912.02975* (2019).

References III

Haohan Wang et al. *High Frequency Component Helps Explain the Generalization of Convolutional Neural Networks*. 2019. arXiv: 1905.13545 [cs.CV].

Ilya Kostrikov, Denis Yarats, and Rob Fergus. *Image Augmentation Is All You Need: Regularizing Deep Reinforcement Learning from Pixels*. 2020. arXiv: 2004.13649 [cs.LG].

Michael Laskin et al. "Reinforcement Learning with Augmented Data". In: *arXiv preprint arXiv:2004.14990* (2020).

Bhairav Mehta et al. "Active domain randomization". In: *Conference on Robot Learning*. 2020, pp. 1162–1176.

Roberta Raileanu et al. "Automatic Data Augmentation for Generalization in Deep Reinforcement Learning". In: *arXiv preprint arXiv:2006.12862* (2020).

References IV

Rui Wang et al. *Enhanced POET: Open-Ended Reinforcement Learning through Unbounded Invention of Learning Challenges and their Solutions*. 2020. arXiv: 2003.08536 [cs.NE].

Zhaoming Xie et al. "ALLSTEPS: Curriculum-driven Learning of Stepping Stone Skills". In: 2020. arXiv: 2005.04323 [cs.GR].

Chang Ye et al. "Rotation, Translation, and Cropping for Zero-Shot Generalization". In: *arXiv preprint arXiv:2001.09908* (2020).