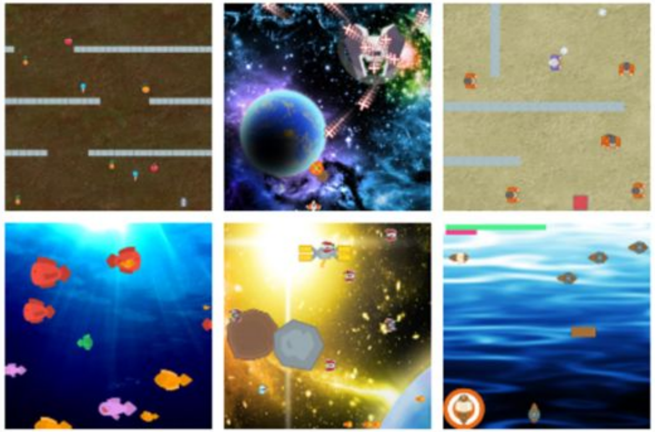
UC San Diego

Improving Policy Optimization with Generalist-Specialist Learning

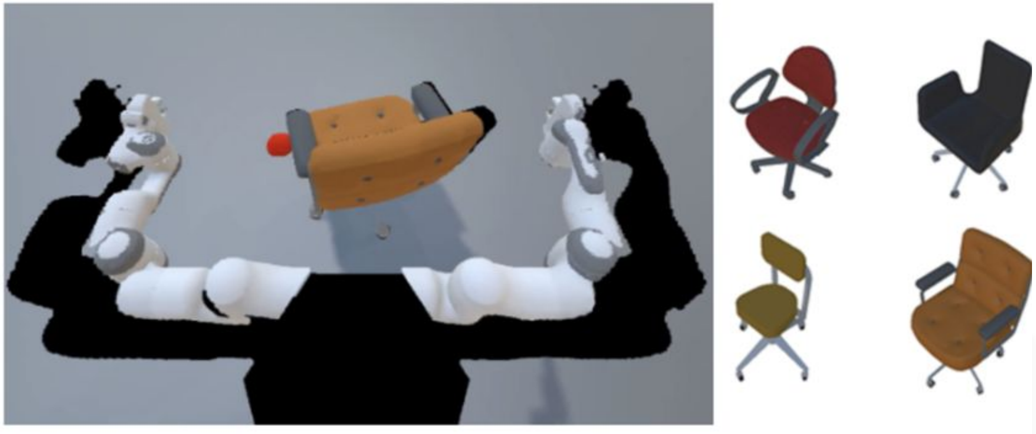
Zhiwei Jia, Xuanlin Li, Zhan Ling, Shuang Liu, Yiran Wu, Hao Su
University of California, San Diego

Background & Motivations

- Generalization in RL requires large-scale RL over diverse env. variations
- Large-scale RL is very challenging due to great env. variations



Sampled Tasks from Procgen

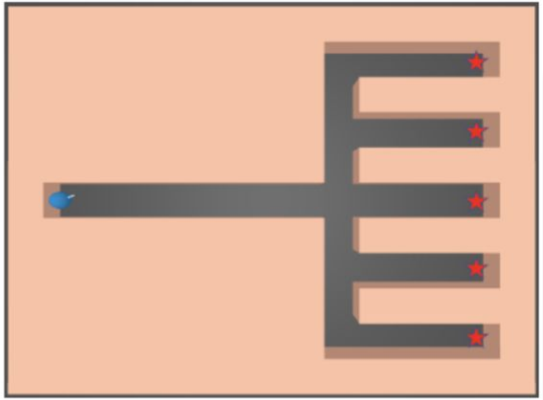


PushChair from ManiSkill

An Illustrative Example

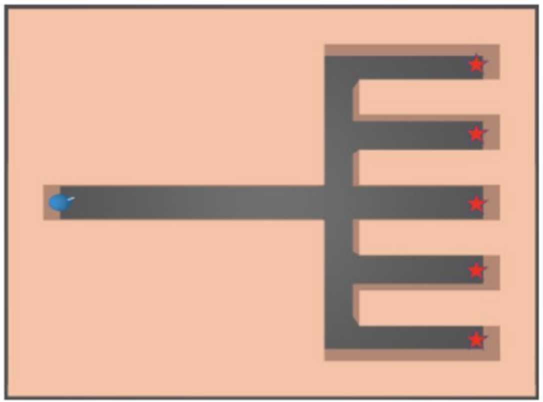
Fork Maze

- The agent (blue, left side) needs to reach goals (red stars, right side)
- Upon each env. reset, only one goal is specified (as a context scalar *c*)
- All env. variations share the same common path at the beginning



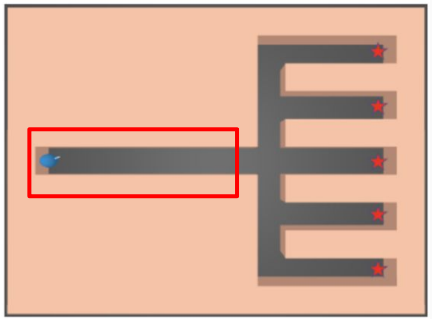
Limitation of Generalists

- Train a single agent (a generalist) jointly on all env. variations (i.e., the reach all of the goals given the context *c*)



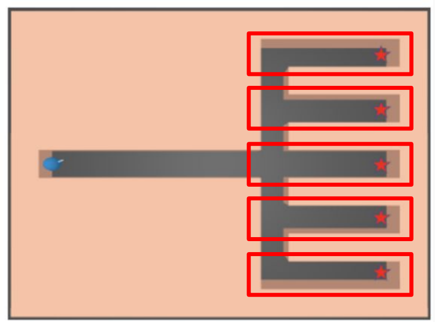
Limitation of Generalists

- Generalists suffer from **catastrophic ignorance** & **catastrophic forgetting**
  - the agent ignores context c and fails to distinguish between different goals, since the c plays little role in the early stages of learning.
  - the agent struggles to learn to solve the environment variations altogether in later stages due to difficulty of memorization for NNs.



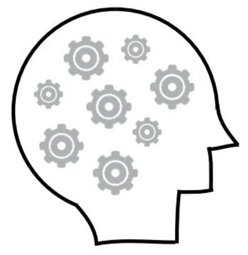
Limitation of Generalists

- Alternatively, we can train a bunch of agents, each handling only a subset of environment variations (**specialists**)
- Specialists avoid the aforementioned issues but also come with a cost



Key Observations: Generalists vs. Specialists

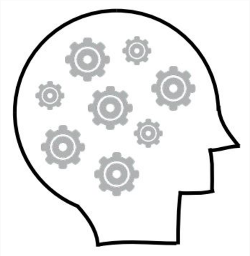
- Generalists learn
  - faster (more sample efficient) at the beginning
- Specialists learn
  - slower (less sample efficient) at the beginning



vs.

Key Observations: Generalists vs. Specialists

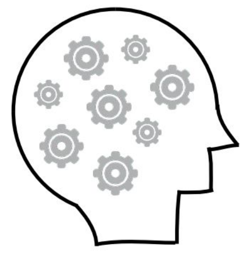
- Generalists learn
  - **faster** (more sample efficient) at the beginning
  - but **worse** performance in later stages
- Specialists learn
  - **slower** (less sample efficient) at the beginning
  - but **better** performance in later stages



vs.

Key Observations: Generalists vs. Specialists

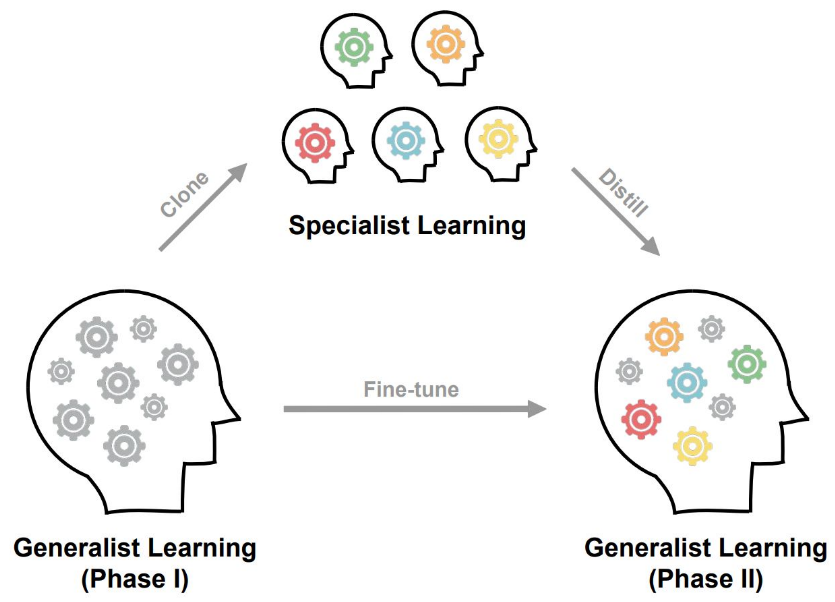
- Generalists learn
  - **faster** (more sample efficient) at the beginning
  - but **worse** performance in later stages
- Specialists learn
  - **slower** (less sample efficient) at the beginning
  - but **better** performance in later stages
  - In addition, can train specialists in parallel (**faster** in wall time)



vs.

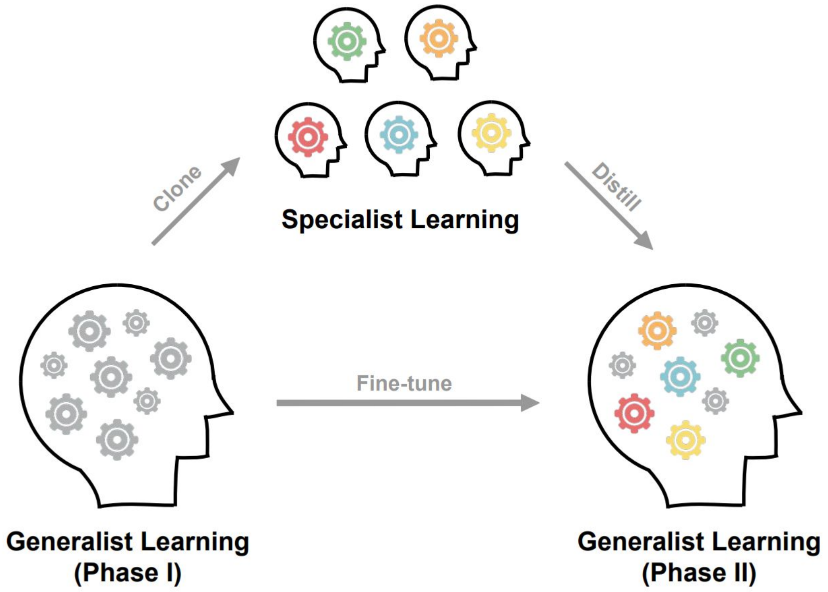
Generalist-Specialist Learning (GSL) - A Meta-Algorithm

1. Train a **generalist** G on all training env. variations.



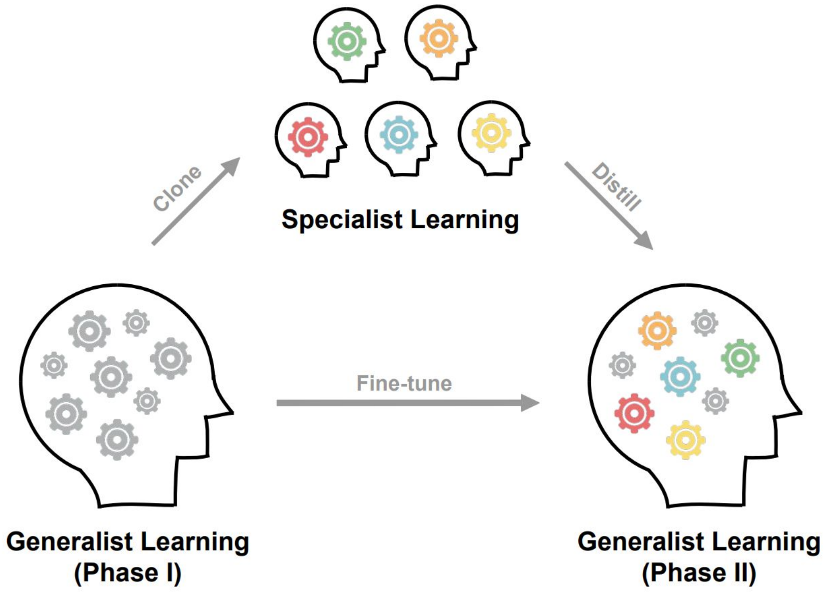
Generalist-Specialist Learning (GSL) - A Meta-Algorithm

1. Train a **generalist** G on all training env. variations.
2. Stop when it plateaus according some criterion *H*



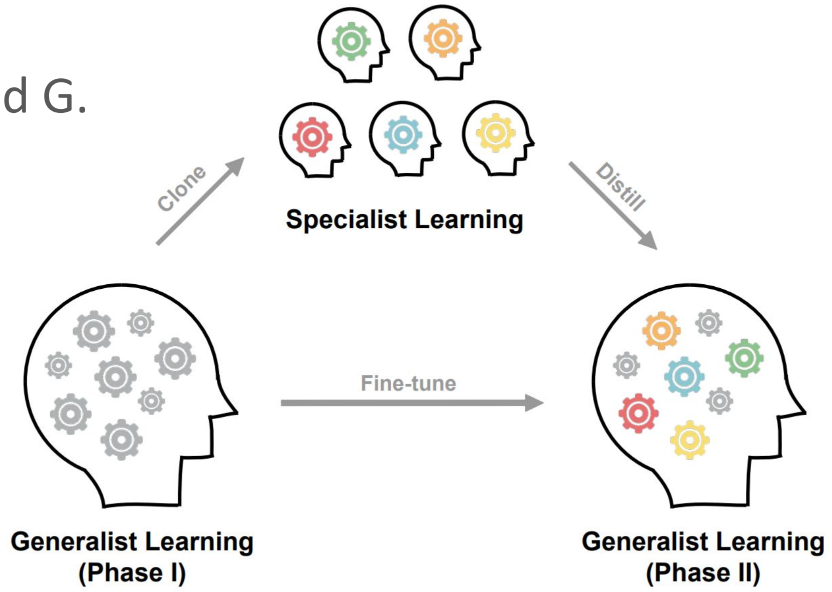
Generalist-Specialist Learning (GSL) - A Meta-Algorithm

1. Train a **generalist** G on all training env. variations.
2. Stop when it plateaus according some criterion *H*
    a. Train a bunch of **specialists** {S}, each loaded from the checkpoints of G and only handles a selective subset of all training env. variations.



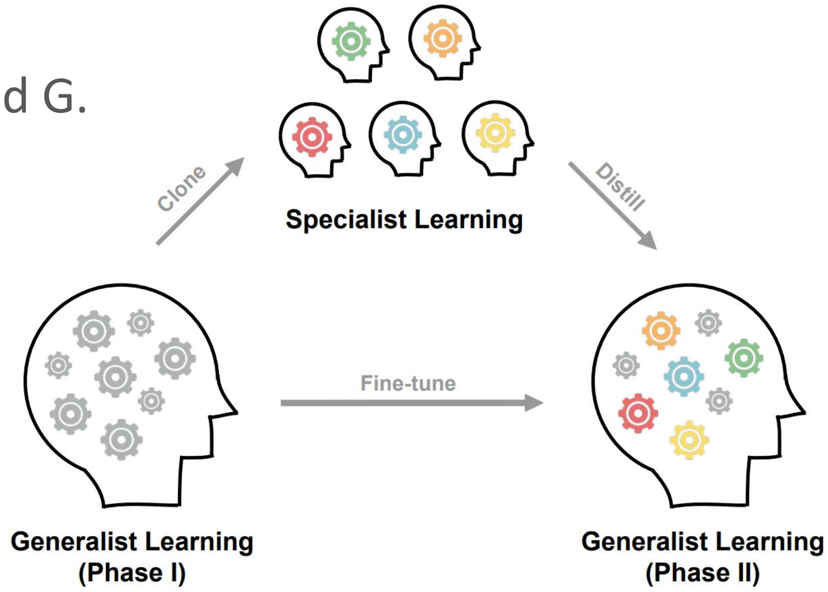
Generalist-Specialist Learning (GSL) - A Meta-Algorithm

1. Train a **generalist** G on all training env. variations.
2. Stop when it plateaus according some criterion *H*
    a. Train a bunch of **specialists** {S}, each loaded from the checkpoints of G and only handles a selective subset of all training env. variations.
    b. Collect demonstrations from the {S} and G.



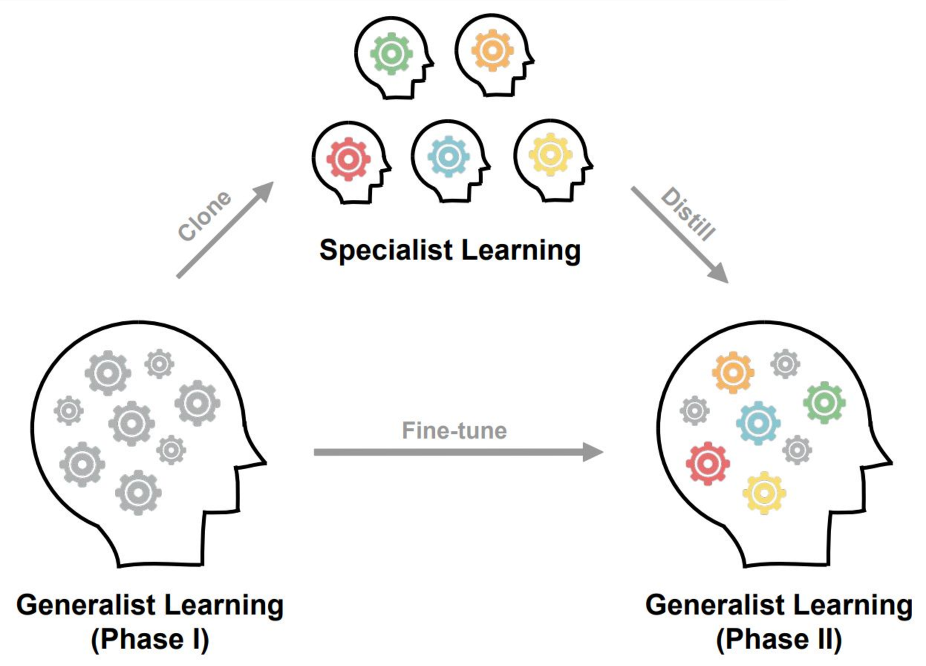
Generalist-Specialist Learning (GSL) - A Meta-Algorithm

1. Train a **generalist** G on all training env. variations.
2. Stop when it plateaus according some criterion *H*
   a. Train a bunch of **specialists** {S}, each loaded from the checkpoints of G and only handles a selective subset of all training env. variations.
   b. Collect demonstrations from the {S} and G.
3. Fine-tune the **generalist** G with guidance from the collected demos.



Design Choices of GSL

- How to train the specialists?
- When to train the specialists?
- How to fine-tune the generalist?



1. How to Train the Specialists?

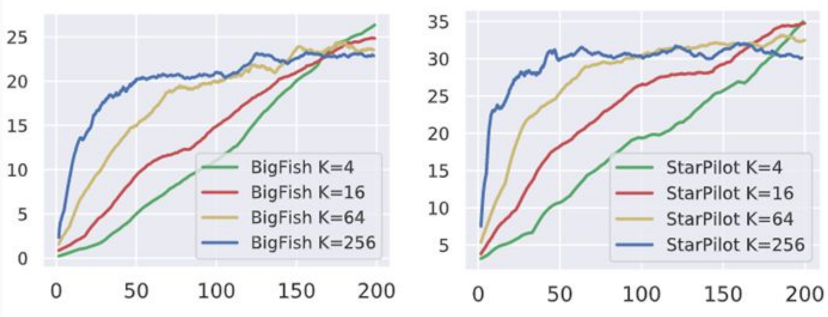
- Only train specialists on the **lowest** performing env. variations because
    - Usually some env. variations are much harder than the others

Training curve of BigFish from Procgen with histograms of avg. rewards across different variations.

1. How to Train the Specialists?

- Only train specialists on the **lowest** performing env. variations because
  - Usually some env. variations are much harder than the others
- Use a **large** population of specialists so that
  - Each specialist can handle less variations →higher return in later stages

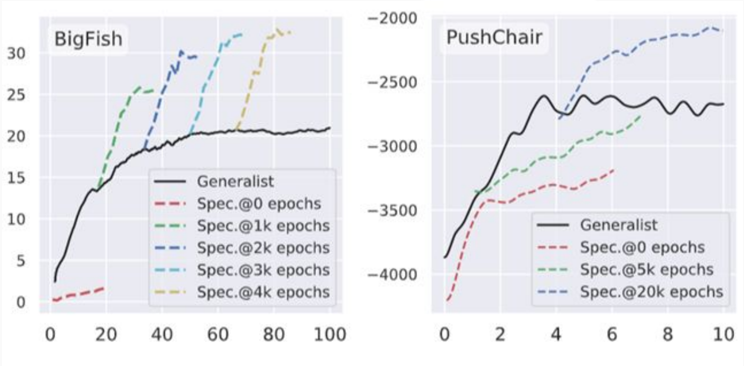
Training curve of BigFish from Procgen with histograms of avg. rewards across different variations.



Training curve of BigFish and StarPilot from Procgen with different number of variations per specialist (K here).

2. When to Train the Specialists?

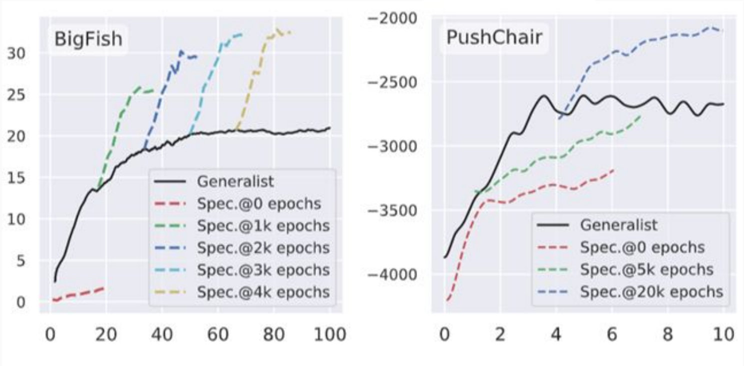
- Only train specialists when the generalist plateaus (to improve efficiency)



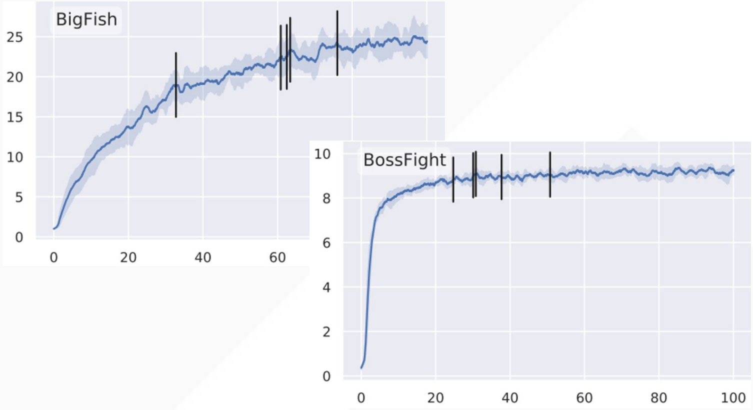
Training curve of specialists starting from different epochs for BigFish from Procgen and PushChair from ManiSkill.

2. When to Train the Specialists?

- Only train specialists when the generalist plateaus (to improve efficiency)
- Plateaus detected by a simple sliding-window style heuristics *H*



Training curve of specialists starting from different epochs for BigFish from Procgen and PushChair from ManiSkill.

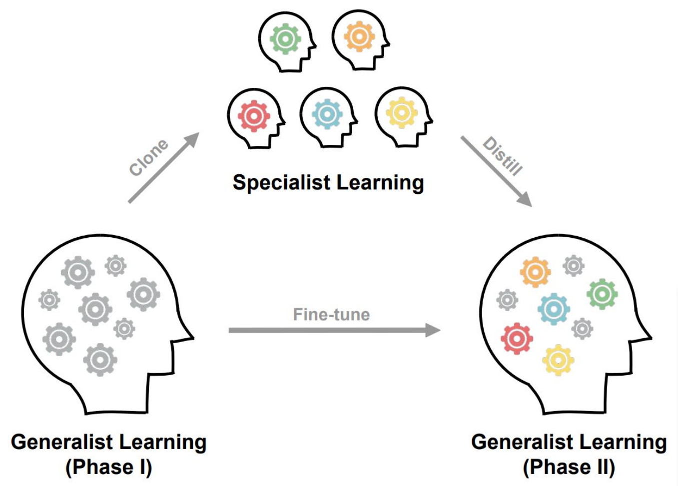


Detected plateaus shown in vertical bars across runs for BigFish and BossFight from Procgen.

3. How to Fine-tune the Generalist?

Why fine-tuning?

- Acquire an agent that performs good on training variations while also generalizes well to others (potentially unseen env. variations)



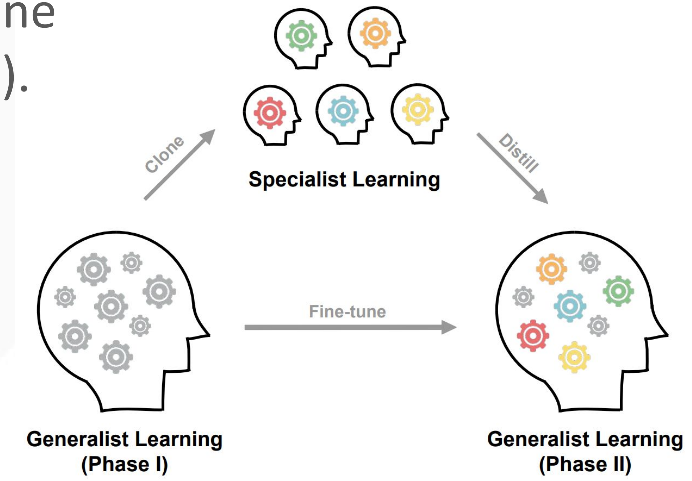
3. How to Fine-tune the Generalist?

Why fine-tuning?

- Acquire an agent that performs good on training variations while also generalizes well to others (potentially unseen env. variations)

Empirical observations

- Online methods (DAPG, GAIL, etc.) utilize online interactions; superior to offline ones (e.g., BC).

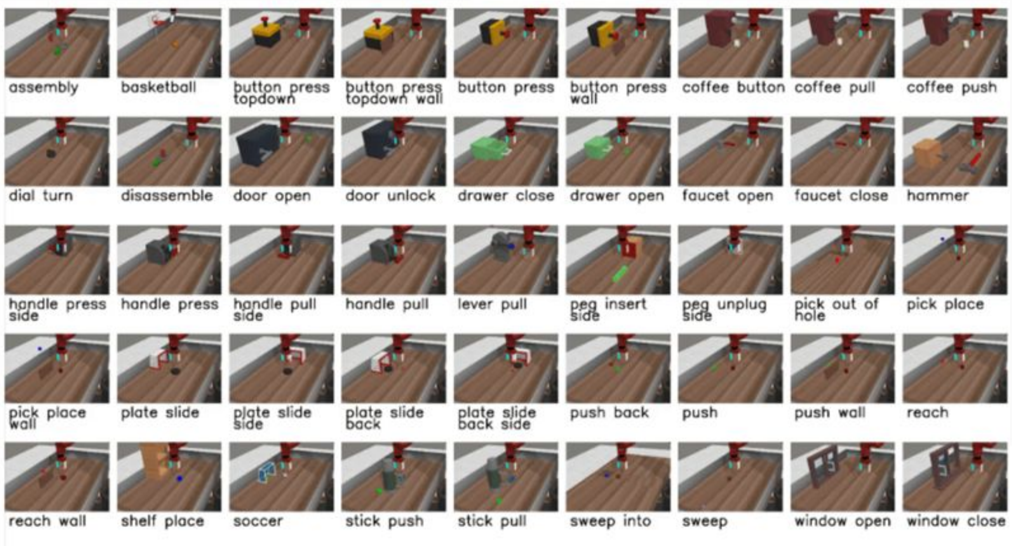


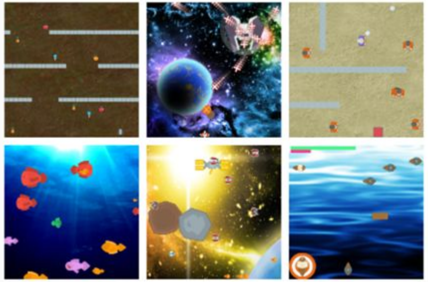
Experiment Setup

Datasets and [backbone RL algorithms]

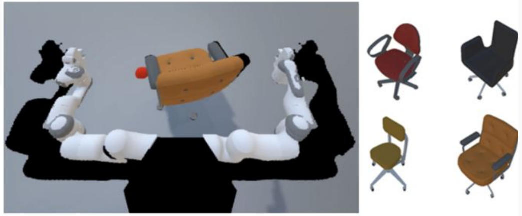
- Procgen [PPO & PPG], Meta-World [PPO], ManiSkill [SAC]

Meta-World



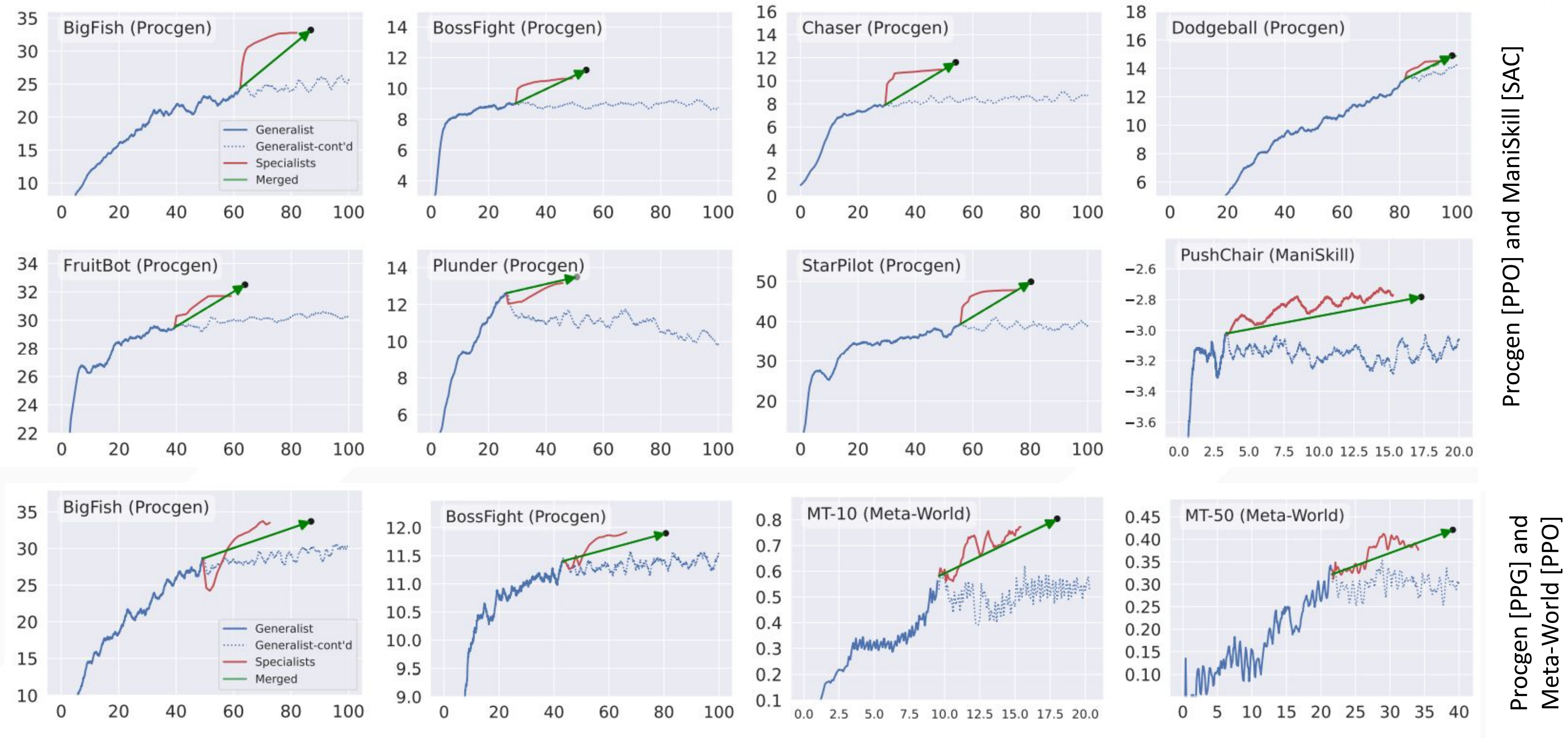


Procgen



ManiSkill

Experiment Results



Experiment Results

	BigFish	BossFight	Chaser	Dodgeball	FruitBot	Plunder	StarPilot
PPO-Train	24.6±0.7	8.6±0.2	8.5±0.3	13.7±0.3	30.1±0.6	10.5±0.8	39.4±1.4
GSL-Train	**31.1±0.8**	**11.3±0.2**	**11.5±0.3**	**15.5±0.2**	**31.9±0.3**	**13.4±0.4**	**49.5±0.4**
PPO-Test	24.3±1.1	8.6±0.3	7.9±0.4	12.7±0.3	29.1±0.5	9.7±0.5	38.0±0.9
GSL-Test	**30.0 ±0.5**	**10.4 ±0.2**	**10.9±0.2**	**14.1±0.3**	**30.5±0.4**	**13.1±0.3**	**48.7±0.5**

	MT-10 (%)	MT-50 (%)
PPO-Train	58.4±10.1	31.1±4.5
GSL-Train	**77.5±2.9**	**43.5±2.2**

	PushChair (k)
SAC-Train	-2.97±2.7
GSL-Train	**-2.78±2.3**

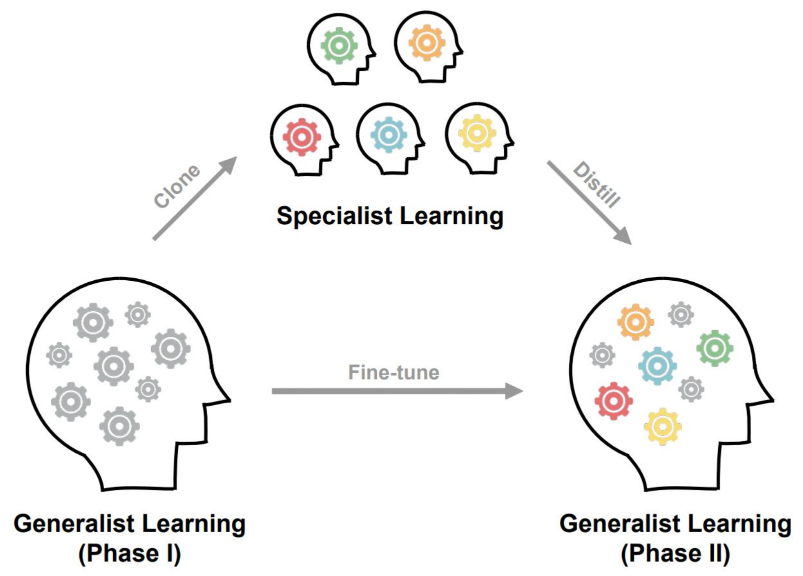
	BigFish	BossFight
PPG-Train	29.4±1.1	11.3±0.2
GSL-Train	**33.5±1.3**	**11.9±0.2**
PPG-Test	28.0±0.9	11.1±0.2
GSL-Test	**30.9 ±0.8**	**11.6±0.2**

Thank You!

Please check our paper for more details

Code available at https://github.com/SeanJia/GSL

Project website at https://zjia.eng.ucsd.edu/gsl



Project Website

Presenter: Zhiwei Jia
Email: zjia@eng.ucsd.edu