

# Fine-tuning Reinforcement Learning Models is Secretly a Forgetting Mitigation Problem

Bartłomiej Cupiał\*



Based on a work co-written with:

Maciej Wołczyk\*, Mateusz Ostaszewski, Michał Bortkiewicz, Michał Zając, Razvan Pascanu, Łukasz Kuciński, Piotr Miłoś

# About me

- **IDEAS NCBR** – leading Polish AI research institution, based in Warsaw
- Part of the ELLIS network
- Many different teams, we work in Sequential Decision Making

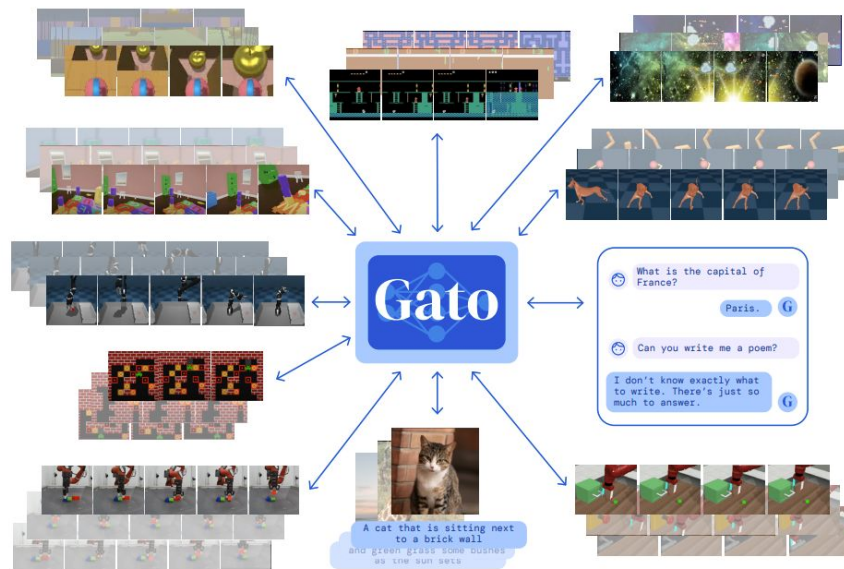
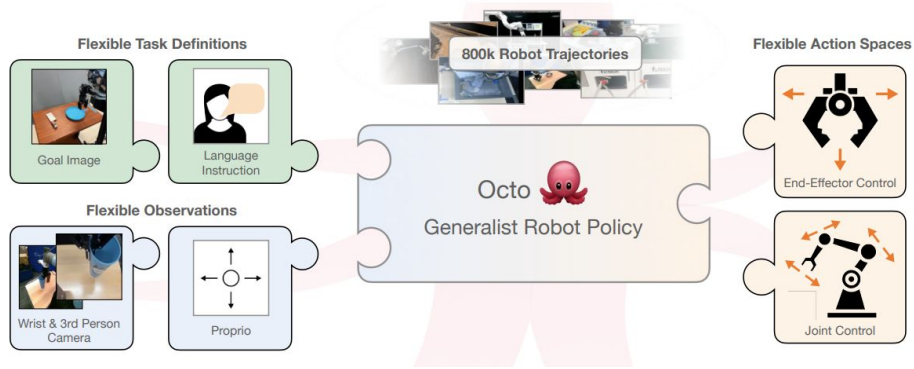


**Bartłomiej Cupiał**

PhD student

- Offline and online RL
- How can we use LLMs in RL?
- LLMs for decision-making

# Motivation - fine-tuning for RL



Offline training scales very well!

# Problems with offline pre-training in RL

- **Upper-bounded by the data-generating expert.**
  - There are methods for improving but they only go so far.
- **We never have enough data.**
- We never clone perfectly and **small errors accumulate.**
  - We might quickly go out-of-distribution.
- Offline methods might misunderstand how their actions impact the environment.

Our proposed solution: **online RL fine-tuning**

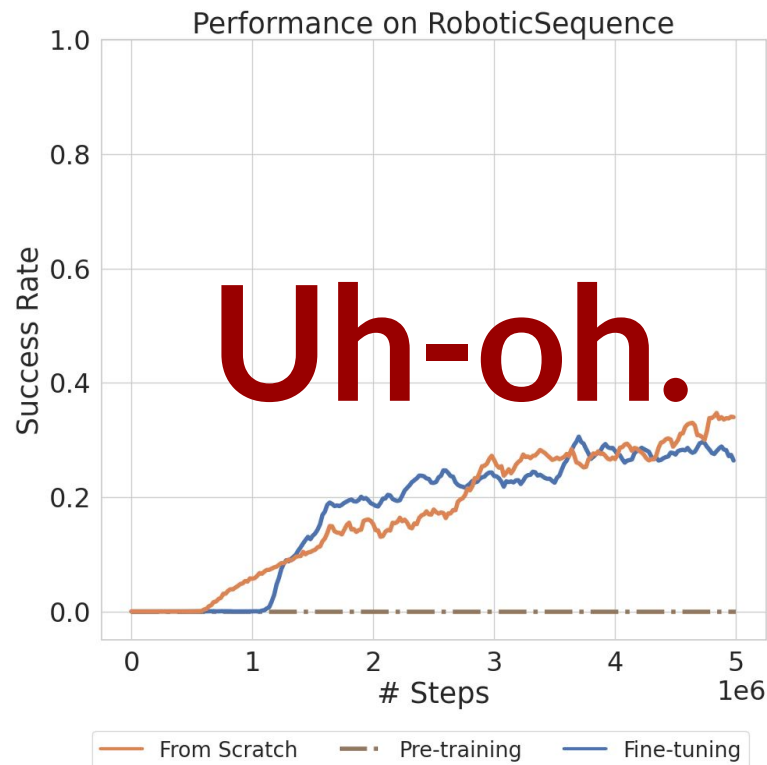
# Testbed: RoboticSequence

- The robot has to perform four tasks in a single episode:
  - Hammer in a nail
  - Push an object
  - Unplug a peg
  - Push an object around a wall
- Episode ends if we do not succeed



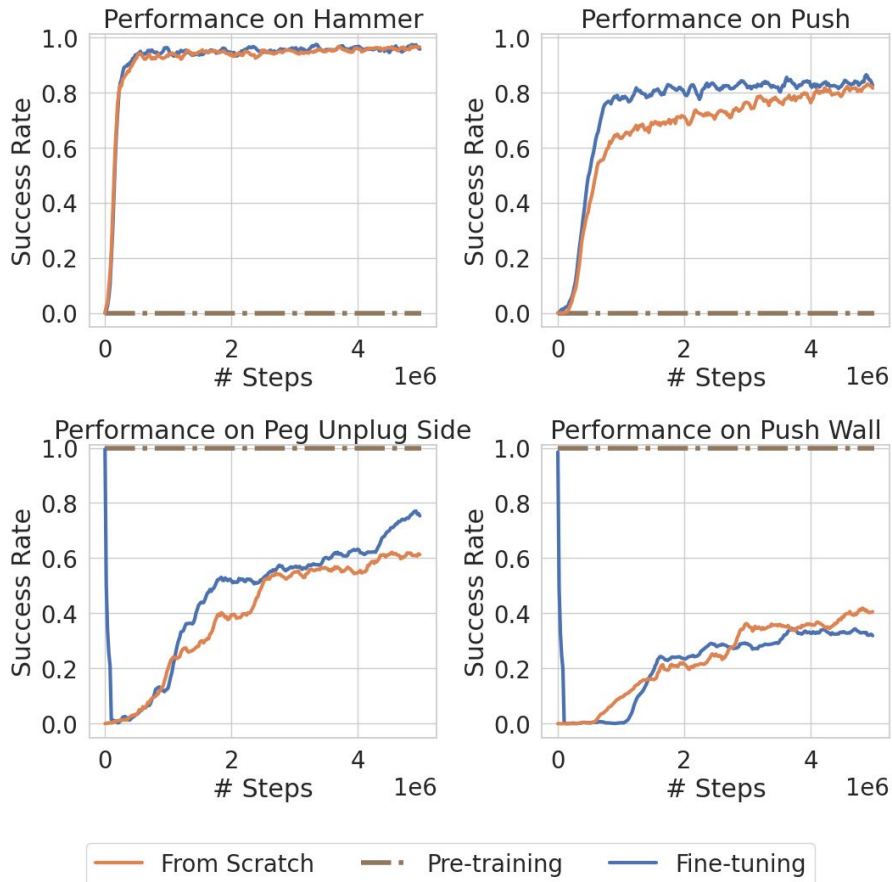
# Fine-tuning scenario

- We have a model that do the last two tasks but not the first two.
- Let's use fine-tune it online to learn the other two tasks!
- Except...



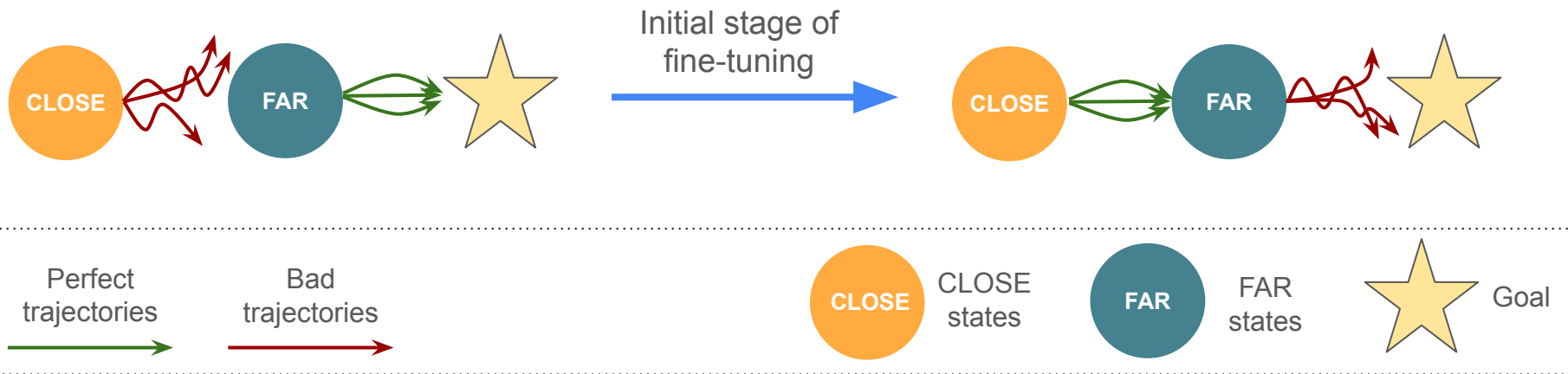
# What happened?

- Let's zoom in on particular tasks.
- **The model forgets the pre-trained knowledge!**



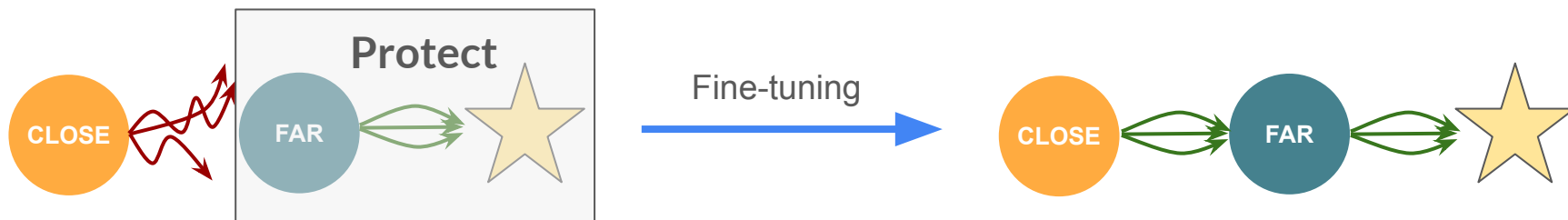
# Why does this happen?

- Let's divide the states in our environment into two sets:
  - **Close** states that are immediately available (e.g., first level)
  - **Far** states that we need to work to get to (e.g., second level)
- In this case, close are the first two tasks that we do not know, and far are tasks that we do know.





We solve this problem, by stopping forgetting on the FAR states.



# We have tools to deal with forgetting!

- The **continual learning** field gives us tools to deal with forgetting.
  - Often it means adding an auxiliary loss to our optimized loss function.
  - Remember the old policy:  $\pi_*$  with parameters  $\theta_*$  which is good on **Far**
- **Episodic Memory (EM)** – add examples from **Far** in the buffer
- **Elastic Weight Consolidation (EWC)**

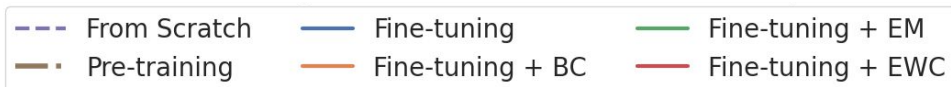
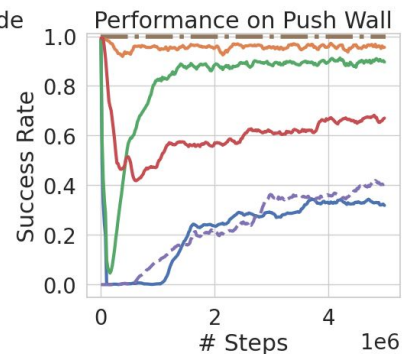
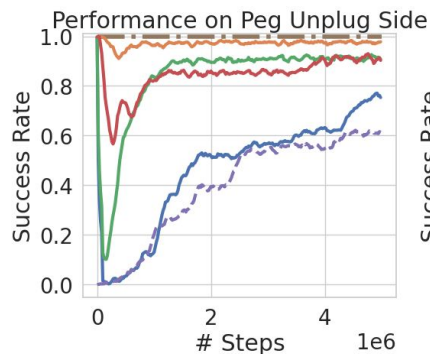
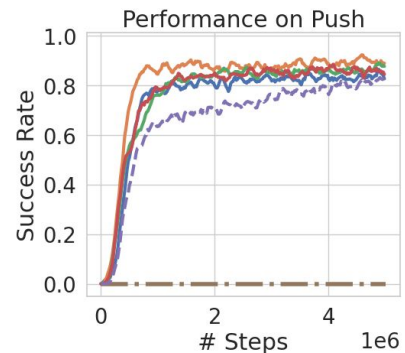
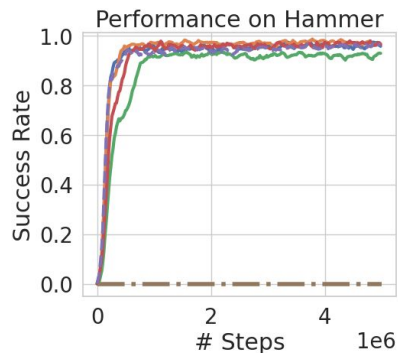
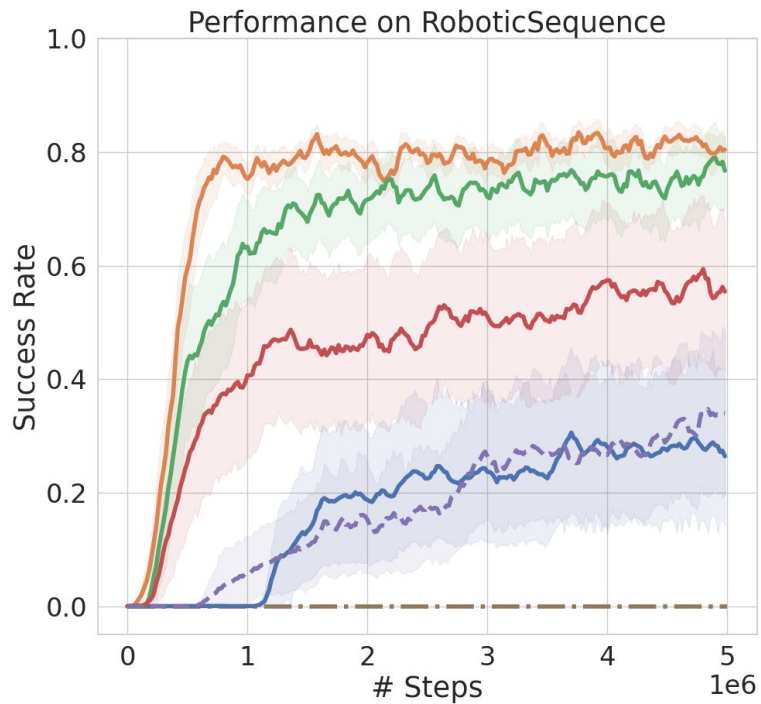
$$\mathcal{L}_{EWC} = \sum_i F^i (\theta_*^i - \theta^i)^2$$

- **Behavioral Cloning (BC)**

$$\mathcal{L}_{BC}(\theta) = \mathbb{E}_{s \sim \mathcal{B}} [D_{KL}^s(\pi_*(\cdot|s) \parallel \pi_\theta(\cdot|s))]$$

- **Kickstarting (KS)** - same as BC, but sample from  $\pi$  rather than  $\mathcal{B}$

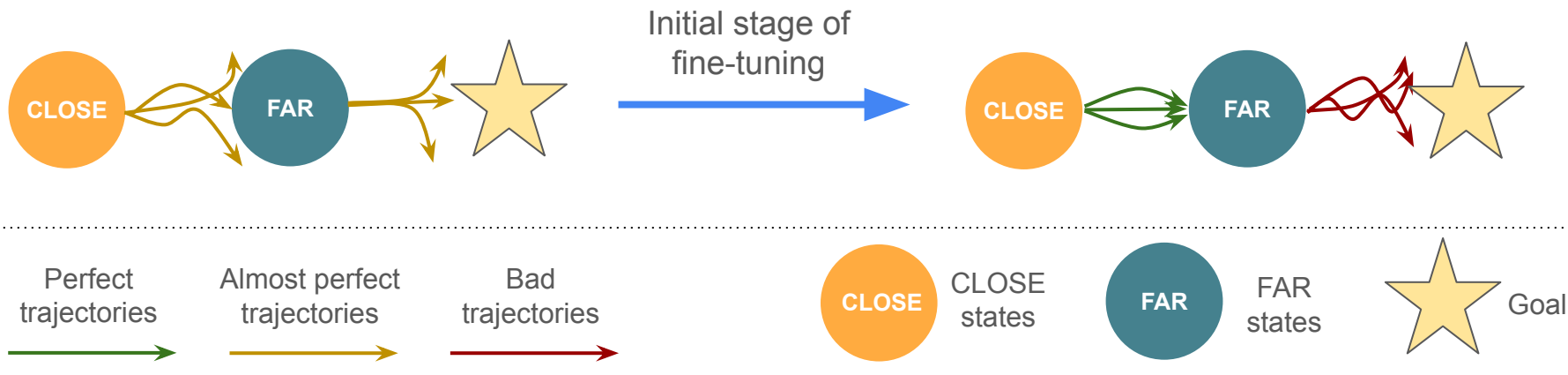
# Fine-tuning, Round 2: now with knowledge retention



# Okay, but what if there's no environment shift?

- We might see forgetting even if we pre-train and finetune on the same env:

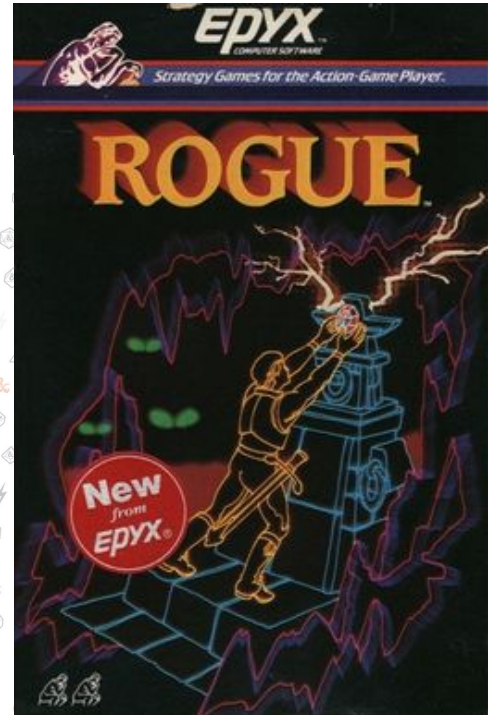
**Pre-train offline and fine-tune online.**



- Behavioral cloning is not perfect, so the fine-tuned model will err and see **Far** states less often than during pre-training.
- **Limited exposure to states we previously trained on leads to forgetting.**

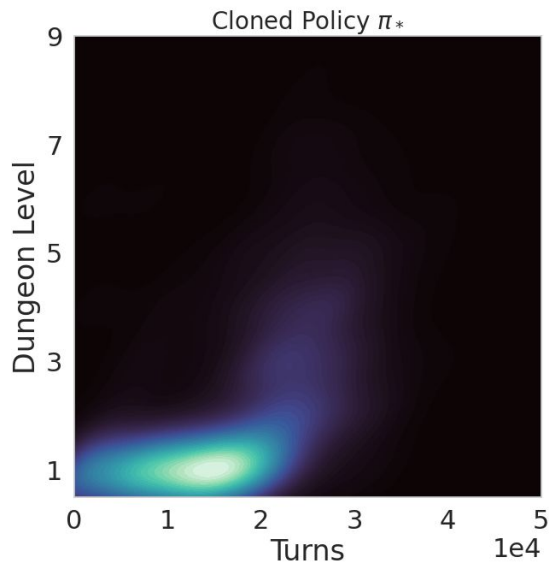
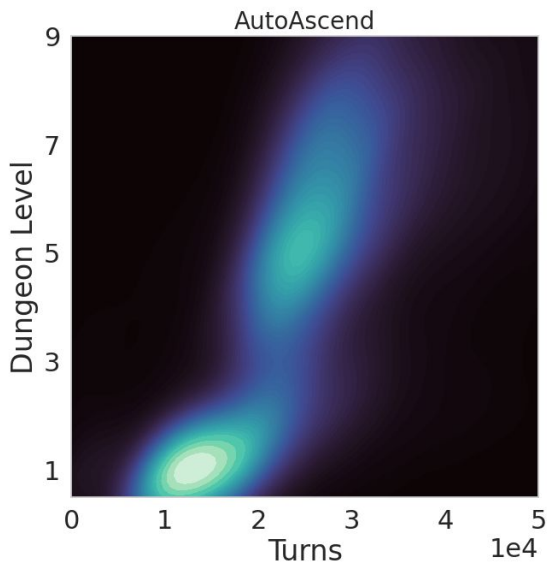
# NetHack - next level

- Single player fantasy game from 1987
- Rogue-like game taking inspiration from rogue
- Grid world with rules from dungeons and dragons



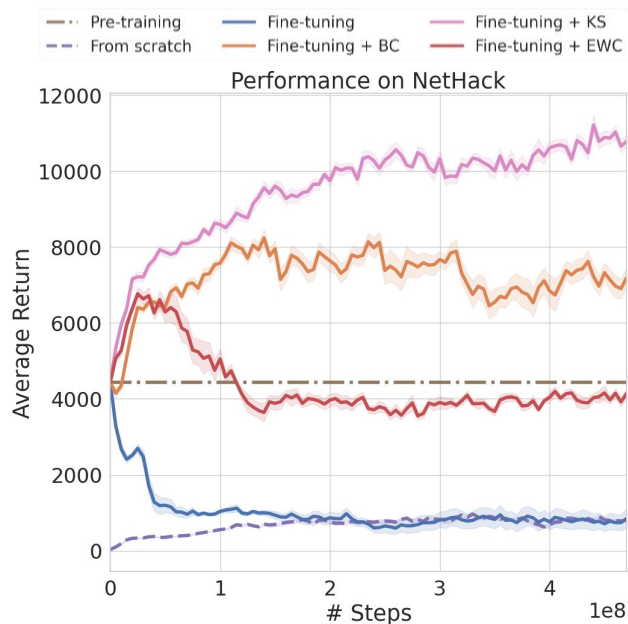
# Distribution shift in the pre-trained model

- BC agent struggles to visit deeper parts of the dungeon



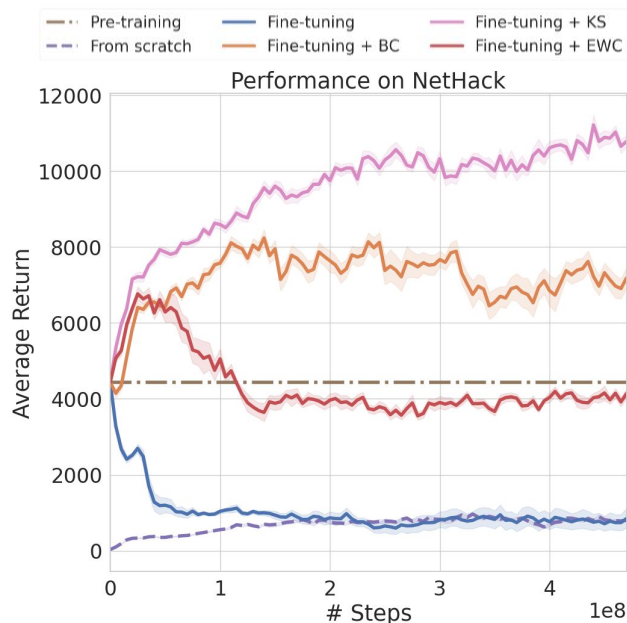
# Main Results

- Vanilla fine-tuning degenerates to the level of model trained from scratch,
- Choosing an effective method for knowledge retention is nuanced,
- Kickstarting managed to double the performance of pre-trained model achieving 10K points!



# Main Results

- Our results demonstrate effectiveness of pre-training and fine-tuning, achieving state-of-the-art results!

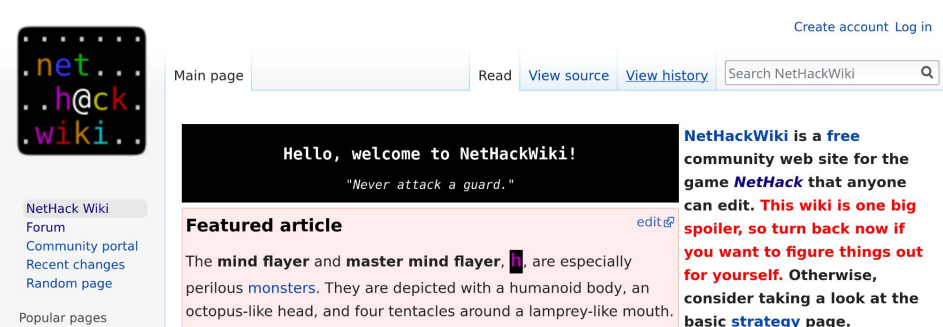


Models	Human Monk
<b>Offline only</b>	
DQN-Offline (Hambro et al., 2022b)	$0.0 \pm 0.0$
CQL (Hambro et al., 2022b)	$366 \pm 35$
IQL (Hambro et al., 2022b)	$267 \pm 28$
BC (CDGPT5) (Hambro et al., 2022b;a)	$1059 \pm 159$
Scaled-BC (Tuyts et al., 2023)	$5218 \pm -$
<b>Offline + Online</b>	
From Scratch + KS (Hambro et al., 2022b)	$2090 \pm 123$
From Scratch + BC (Hambro et al., 2022b)	$2809 \pm 103$
LDD* (Mu et al., 2022)	$2100 \pm -$
<b>Scaled-BC + Fine-tuning + KS (ours)</b>	<b><math>10588 \pm 672</math></b>

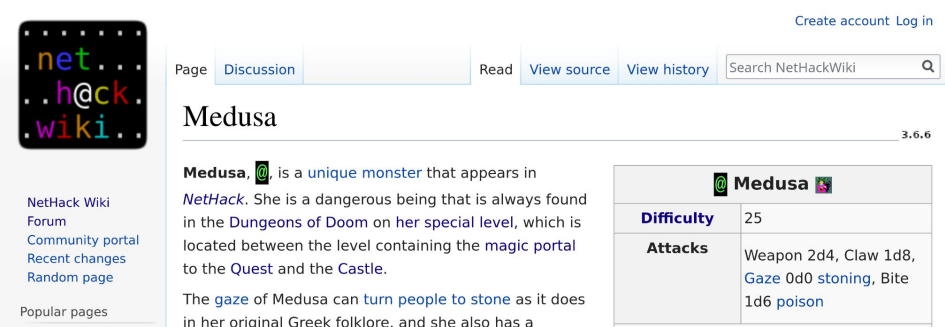


# Future steps!

- **GOAL:** Beat *AutoAscend*. In particular discover new strategies / behaviors that aren't present in the hardcoded bot.
- Combine LLMs with RL:
  - LLMs can reason and leverage information from NetHackWiki,
  - How to connect language space and actions?
  - Can we use data from NethackWiki to suggest new strategies?
  - Learning from Knowledge Bases: 3000+ articles on the NetHack Wiki.



The screenshot shows the main page of NetHack Wiki. At the top right, there are links for "Create account" and "Log in". Below this is a navigation bar with "Main page", "Read", "View source", and "View history" buttons, followed by a search box labeled "Search NetHackWiki". A large black banner with white text reads "Hello, welcome to NetHackWiki!" and "Never attack a guard." Below the banner is a "Featured article" section with a red header. The article text describes "mind flyer" and "master mind flyer" monsters. To the right of the featured article is a blue box with white text: "NetHackWiki is a free community web site for the game NetHack that anyone can edit. This wiki is one big spoiler, so turn back now if you want to figure things out for yourself. Otherwise, consider taking a look at the basic strategy page." On the left side, there is a sidebar with a "net.hack.wiki" logo and links for "NetHack Wiki", "Forum", "Community portal", "Recent changes", "Random page", and "Popular pages".



The screenshot shows the "Medusa" article page on NetHack Wiki. At the top right, there are links for "Create account" and "Log in". Below this is a navigation bar with "Page", "Discussion", "Read", "View source", and "View history" buttons, followed by a search box labeled "Search NetHackWiki". The article title "Medusa" is prominently displayed. Below the title is a small "net.hack.wiki" logo. The main text of the article describes Medusa as a unique monster in NetHack, located in the Dungeons of Doom. It mentions her special level, her magic portal, and her gaze ability. To the right of the main text is a table with the following data:

Medusa	
<b>Difficulty</b>	25
<b>Attacks</b>	Weapon 2d4, Claw 1d8, Gaze 0d0 stoning, Bite 1d6 poison

Below the table, there is a "Popular pages" section.

# Summary

- We need online fine-tuning to harness the power of RL models.
- Online fine-tuning in RL is plagued by forgetting, but you can use knowledge retention methods to deal with it.
- When done right, fine-tuning gives us very good models:
  - **SOTA in NetHack, doubling the previous record.**
- NetHack remains a very important, challenging environment.

# Thank you for your attention!

- Write to me: [bartlomiej.cupial@gmail.com](mailto:bartlomiej.cupial@gmail.com)
- Twitter / X: @CupiaBart
- Arxiv: <http://arxiv.org/abs/2402.02868>
- Github: <https://github.com/BartekCupial/sample-factory/tree/nethack>