# Proximal Policy Optimization

## Constrained Policy Gradient Methods

Jakub Grzywaczewski
Miłosz Kita
Marta Szuwarska

Warsaw University of Technology

May 9, 2024

# Table of Contents


1. Introduction
2. Related work
   - Q-learning
   - Vanilla gradient method
   - Trust Region Policy Optimization (TRPO)
3. How does PPO work?
4. PPO Results in the Paper
5. PPO in the literature
   - Jack of all Trades
   - RLHF - Reinforcement Learning from Human Feedback
   - Autonomous Robots Dexterity
   - Honorable Mentions
6. Summary
7. Bibliography


# Table of Contents


1. Introduction
2. Related work
   - Q-learning
   - Vanilla gradient method
   - Trust Region Policy Optimization (TRPO)
3. How does PPO work?
4. PPO Results in the Paper
5. PPO in the literature
   - Jack of all Trades
   - RLHF - Reinforcement Learning from Human Feedback
   - Autonomous Robots Dexterity
   - Honorable Mentions
6. Summary
7. Bibliography


# Introduction

1. **PPO Algorithm and Comparison with Other Methods:**
   - Comparative analysis highlighting how PPO offers advantages over other reinforcement learning methods in terms of stability and efficiency.
   - Exploration of the PPO algorithm and its key features.

2. **Real-World Applications of PPO:**
   - Showing various applications of PPO across different industries.

**Objective:**

- Provide a clear understanding of PPO's design.
- Discuss its impact on practical applications.

# Table of Contents


1. Introduction
2. Related work
   - Q-learning
   - Vanilla gradient method
   - Trust Region Policy Optimization (TRPO)
3. How does PPO work?
4. PPO Results in the Paper
5. PPO in the literature
   - Jack of all Trades
   - RLHF - Reinforcement Learning from Human Feedback
   - Autonomous Robots Dexterity
   - Honorable Mentions
6. Summary
7. Bibliography


# Q-learning

**Q-learning** is a model-free reinforcement learning algorithm used to find the optimal action-selection policy for a given finite Markov decision process (MDP).

## Bellman equation for Q-learning

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[ r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

**Downsides of Q-learning are:**

- poorly understood,
- fails on many simple problems.

# Vanilla gradient method

Policy Gradient Methods aim to optimize the policy directly by adjusting the parameters $\theta$ of the policy distribution $\pi_\theta$ in a way that maximizes the expected reward.

## Policy Gradient Estimator

In Gradient methods, the most typically used estimator is the expected finite-horizon undiscounted return of the policy:

$$\hat{g} = \hat{\mathbb{E}}_t \left[ \nabla_\theta \log \pi_\theta(a_t|s_t) \hat{A}_t \right],$$

where $\hat{A}$ is an estimator of the advantage function at time-step t.

- The goal is to find parameter updates that increase the probability of actions leading to higher rewards.
- It focuses on evaluating actions that perform better than the average, guiding the learning process toward optimal decisions.

# Vanilla gradient method

**Downsides of Vanilla Policy Gradient are:**

- poor data efficiency,
- lack of robustness,
- destructively large policy updates when performing multiple batch updates.

# Trust Region Policy Optimization (TRPO)

In TRPO[6] the objective function is subject to the constrained on the policy update size.

$$\arg\max_{\theta} \ \hat{\mathbb{E}}_t \left[ \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t \right] \tag{1}$$

$$\text{subject to } \hat{\mathbb{E}}_t \left[ KL[\pi_\theta(\cdot|s_t), \pi_{\theta_{old}}(\cdot|s_t)] \right] \leq \delta \tag{2}$$

To optimize this objective TRPO requires second-order optimization. TRPO allows us to overcome the downside of the Vanilla Gradient Method and perform multiple batch updates to the policy. This is possible due to the use of **Importance Sampling** and this **KL Divergence constraint**.

# Importance Sampling in TRPO

In all policy gradient methods we are alternating between data collection and network optimization. Therefor if we were to perform multiple updates to the network during training we have an off-policy method. To fix that we use **Importance Sampling**.

$$\mathbb{E}_{\pi_\theta}\left[A(a,s)\right] = \iint_{\mathcal{A}\times\mathcal{S}} \pi_\theta(a|s) A(a,s)\, da\, ds \tag{3}$$

$$= \iint_{\mathcal{A}\times\mathcal{S}} \pi_{\theta_{old}}(a|s) \frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} A(a,s)\, da\, ds \tag{4}$$

$$= \mathbb{E}_{\pi_{\theta_{old}}}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} A(a,s)\right] \tag{5}$$

If the $\pi_{\theta_{old}}$ policy is the one generating the data then we can set an objective to maximise the following function for all timesteps $t$:

$$\mathbb{E}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} A_t\right]$$

# TRPO downsides

**Downsides of TRPO are:**

- relatively complicated,
- not compatible with architectures that include noise,
- doesn't allow for parameter sharing,
- doesn't allow for easy parallelism.

# Table of Contents


1. Introduction
2. Related work
   - Q-learning
   - Vanilla gradient method
   - Trust Region Policy Optimization (TRPO)
3. How does PPO work?
4. PPO Results in the Paper
5. PPO in the literature
   - Jack of all Trades
   - RLHF - Reinforcement Learning from Human Feedback
   - Autonomous Robots Dexterity
   - Honorable Mentions
6. Summary
7. Bibliography


# Proximal Policy Optimization

Proximal Policy Optimization (PPO) [5] takes a simpler objective then the one proposed in TRPO. TRPO maximises:

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t \left[ \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t \right] = \hat{\mathbb{E}}_t \left[ r_t(\theta) \hat{A}_t \right], \tag{7}$$

constrained by the KL divergence. The PPO paper proposes two alternatives:

- $L^{CLIP}$ - Clipped Surrogate Objective,
- $L^{KLPEN}$ - Adaptive KL Penalty Objective.

We will focus on the $L^{CLIP}$.

# Clipped Surrogate Objective

The Clipped Surrogate Objective is defined as follows:

$$L^{CLIP} = \hat{\mathbb{E}}_t \left[ \min \left( r_t(\theta)\hat{A}_t \ , \ clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right]. \tag{8}$$

| | $p_t(\theta) > 0$ | $A_t$ | **Return Value of *min*** | **Objective is Clipped** | **Sign of Objective** | **Gradient** |
|---|---|---|---|---|---|---|
| **1** | $p_t(\theta) \in [1-\epsilon, 1+\epsilon]$ | $+$ | $p_t(\theta)A_t$ | no | $+$ | ✓ |
| **2** | $p_t(\theta) \in [1-\epsilon, 1+\epsilon]$ | $-$ | $p_t(\theta)A_t$ | no | $-$ | ✓ |
| **3** | $p_t(\theta) < 1-\epsilon$ | $+$ | $p_t(\theta)A_t$ | no | $+$ | ✓ |
| **4** | $p_t(\theta) < 1-\epsilon$ | $-$ | $(1-\epsilon)A_t$ | yes | $-$ | **0** |
| **5** | $p_t(\theta) > 1+\epsilon$ | $+$ | $(1+\epsilon)A_t$ | yes | $+$ | **0** |
| **6** | $p_t(\theta) > 1+\epsilon$ | $-$ | $p_t(\theta)A_t$ | no | $-$ | ✓ |

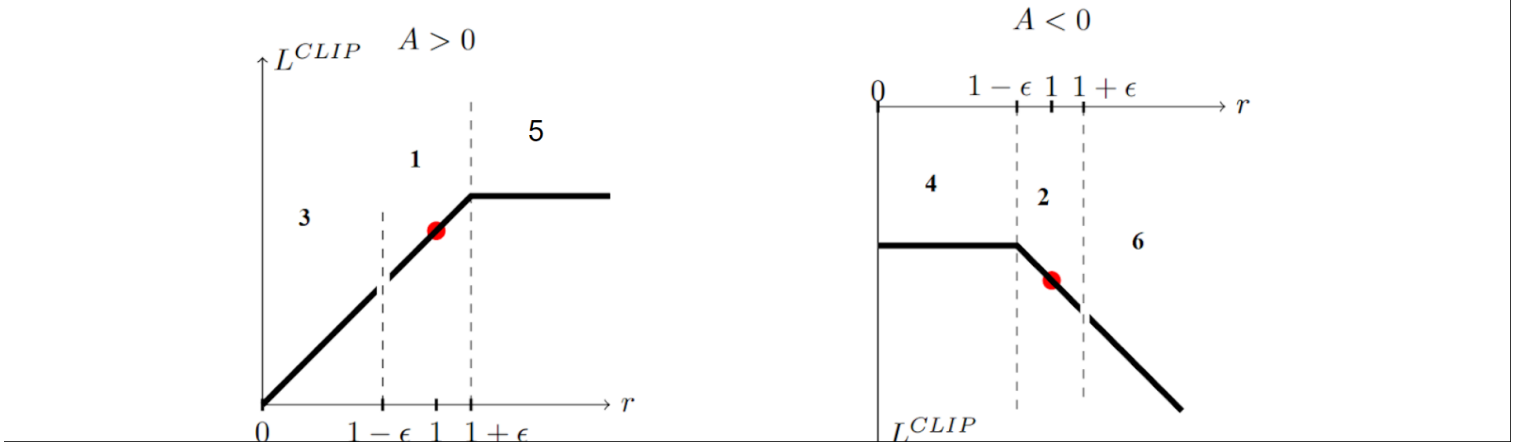



Figure: Clipping table found on Hugging Face taken from the paper [1]

# Comparison to unconstrained objective

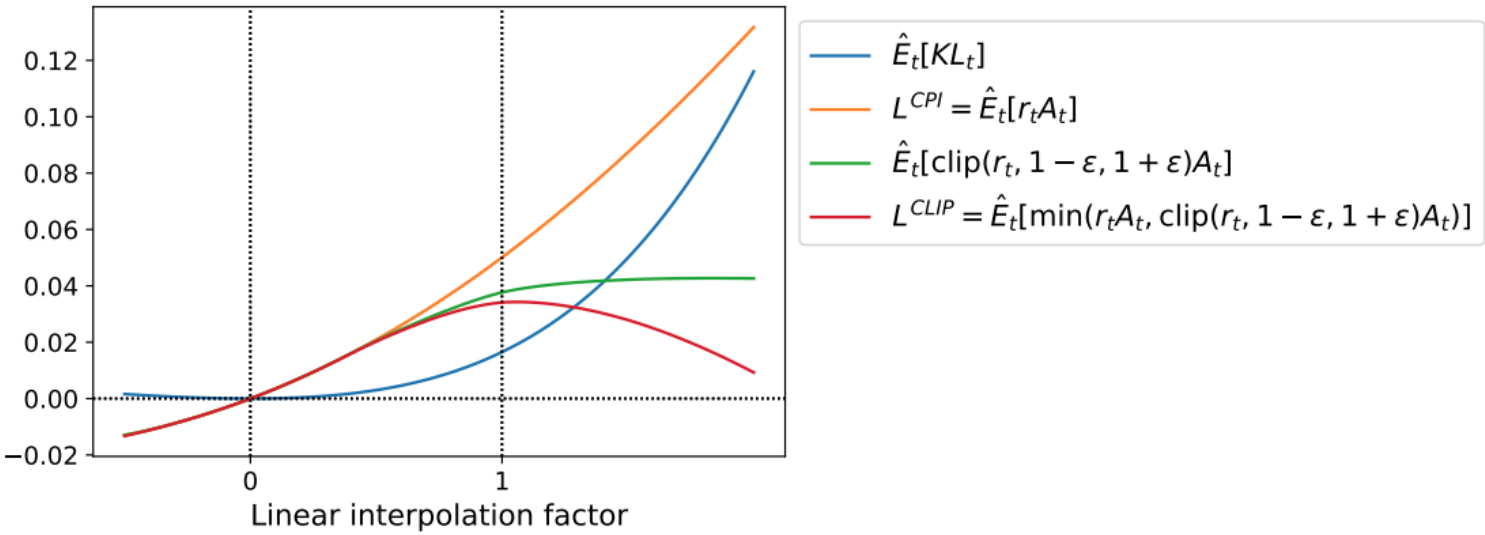



Figure: Surrogate objectives, as they interpolated between the initial policy parameter $\theta_{old}$, and the updated policy parameter, which they computed after one iteration of PPO. Taken from the paper [5]

# Advantage estimator

Advantage is defined as:

$$A(a, s) = Q(a, s) - V(s), \tag{9}$$

where $Q(a, s)$ is an Action State Value and the $V$ is a State Value.

PPO uses an estimate of that advantage function by training an State Value model $V(s)$ along side the policy:

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1}, \tag{10}$$

$$\text{where} \quad \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{11}$$

# Shared parameters problem

"If using a neural network architecture that shares parameters between the policy and value function, we must use a loss function that combines the policy surrogate and a value function error term"[5]

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t \left[ L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t) \right],$$

where $c_1, c_2$ are the coefficients, the $S$ denotes the entropy bonus and the $L_t^{VF}$ is a squared-error loss.

Typically the $c_1$ coefficient for shared parameters network is set to 1.

# PPO Actor-Critic

**Algorithm 1** PPO, Actor-Critic Style

**for** iteration=$1, 2, \ldots$ **do**
  **for** actor=$1, 2, \ldots, N$ **do**
    Run policy $\pi_{\theta_{\text{old}}}$ in environment for $T$ timesteps
    Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
  **end for**
  Optimize surrogate $L$ wrt $\theta$, with $K$ epochs and minibatch size $M \leq NT$
  $\theta_{\text{old}} \leftarrow \theta$
**end for**

Figure: Example of an Actor-Critic style algorithm presented in the paper [5]

# Table of Contents


1. Introduction
2. Related work
   - Q-learning
   - Vanilla gradient method
   - Trust Region Policy Optimization (TRPO)
3. How does PPO work?
4. PPO Results in the Paper
5. PPO in the literature
   - Jack of all Trades
   - RLHF - Reinforcement Learning from Human Feedback
   - Autonomous Robots Dexterity
   - Honorable Mentions
6. Summary
7. Bibliography


# Comparison of Surrogate Objectives [5]

| algorithm | avg. normalized score |
|---|---|
| No clipping or penalty | -0.39 |
| Clipping, $\epsilon = 0.1$ | 0.76 |
| **Clipping, $\epsilon = 0.2$** | **0.82** |
| Clipping, $\epsilon = 0.3$ | 0.70 |
| Adaptive KL $d_{\text{targ}} = 0.003$ | 0.68 |
| Adaptive KL $d_{\text{targ}} = 0.01$ | 0.74 |
| Adaptive KL $d_{\text{targ}} = 0.03$ | 0.71 |
| Fixed KL, $\beta = 0.3$ | 0.62 |
| Fixed KL, $\beta = 1.$ | 0.71 |
| Fixed KL, $\beta = 3.$ | 0.72 |
| Fixed KL, $\beta = 10.$ | 0.69 |

Figure: Results from continuous control benchmark. Average normalized scores for each algorithm/hyperparameter setting.

# PPO Results for MuJoCo Environments [5]

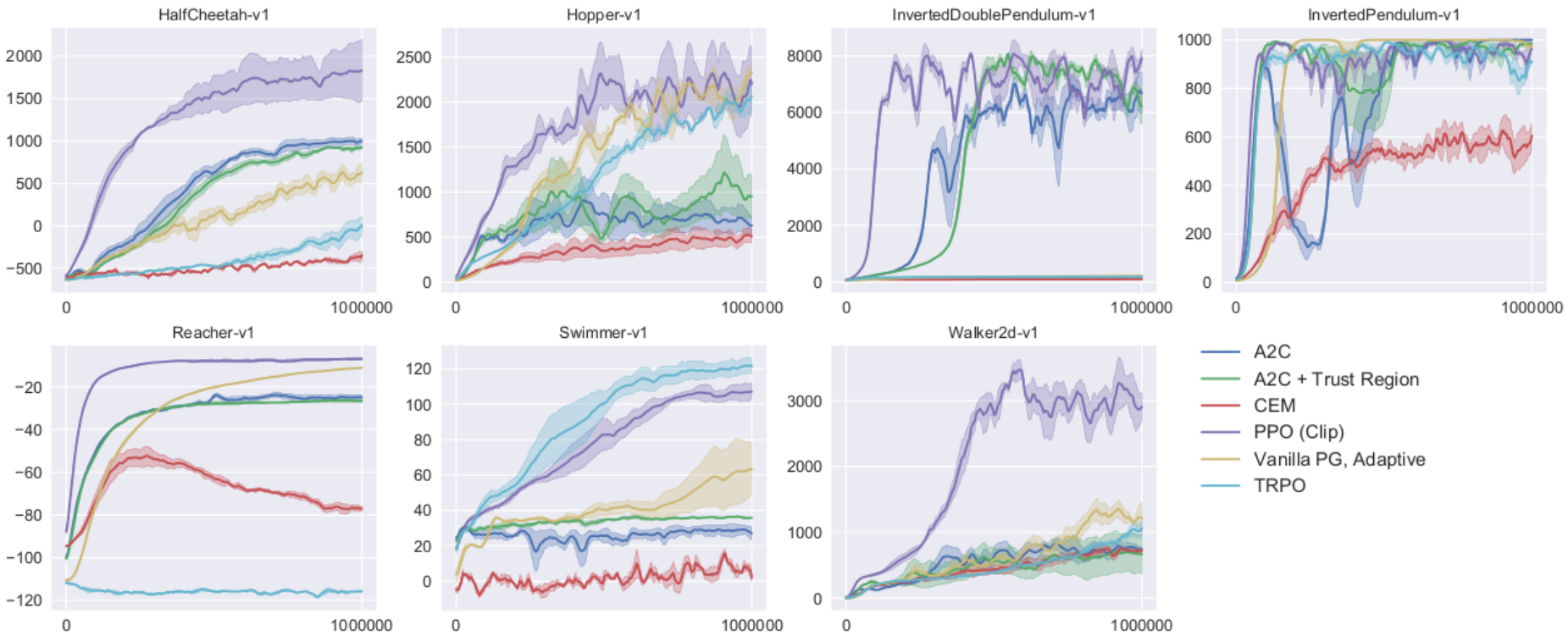



Figure: Comparison of several algorithms on several MuJoCo environments, training for one million timesteps.

# PPO Results for Roboschool Tasks [5]

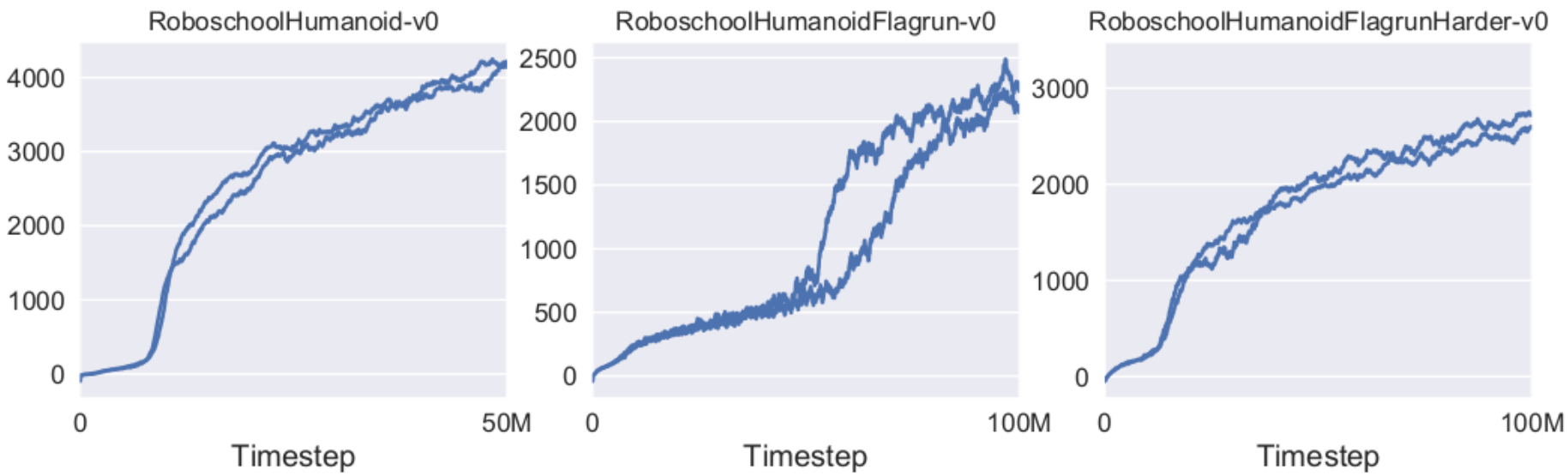



Figure: Learning curves from PPO on 3D humanoid control tasks, using Roboschool.

# PPO Results for Atari Games [5]

| | A2C | ACER | PPO | Tie |
|---|---|---|---|---|
| (1) avg. episode reward over all of training | 1 | 18 | **30** | 0 |
| (2) avg. episode reward over last 100 episodes | 1 | **28** | 19 | 1 |

Figure: Number of Atari games "won" by each algorithm, where the scoring metric is averaged across three trials.

# Table of Contents


1. Introduction
2. Related work
   - Q-learning
   - Vanilla gradient method
   - Trust Region Policy Optimization (TRPO)
3. How does PPO work?
4. PPO Results in the Paper
5. PPO in the literature
   - Jack of all Trades
   - RLHF - Reinforcement Learning from Human Feedback
   - Autonomous Robots Dexterity
   - Honorable Mentions
6. Summary
7. Bibliography


# Jack of all Trades [2]

## Core Challenge

Creating a versatile model that can handle NLP, CV, and RL tasks with a constant set of parameters.

- The model should effectively handle different types of tasks and datasets.
- The model should keep the balance between different domains without any bias.
- The model should adjust to different levels of task complexity.

# JAT Model for Sequential Decision-Making Tasks [2]

- Two embeddings: one for observations with corresponding rewards and another for actions.

## Loss function

$$L = \kappa \cdot L_{obs} + (1 - \kappa) \cdot L_{act} \tag{12}$$

The creators of JAT found the sweet spot for $\kappa \sim 0.005$.

- Asynchronous PPO implemented from Sample Factory
- 8 GPUs and nine days of training on fixed hyperparameters.

# Sample Factory PPO implementation used for JAT [3]

- **Sample Factory** architecture combines a highly efficient, asynchronous, GPU-based sampler with off-policy correction techniques that resulted in the performance of over 100k FPS.
- There are three major computational workloads: environment simulation, model inference, and backpropagation. The key idea behind Sample Factory was to build a system in which the slowest of three workloads never has to wait for any other processes.

# Sample Factory PPO implementation used for JAT [3]

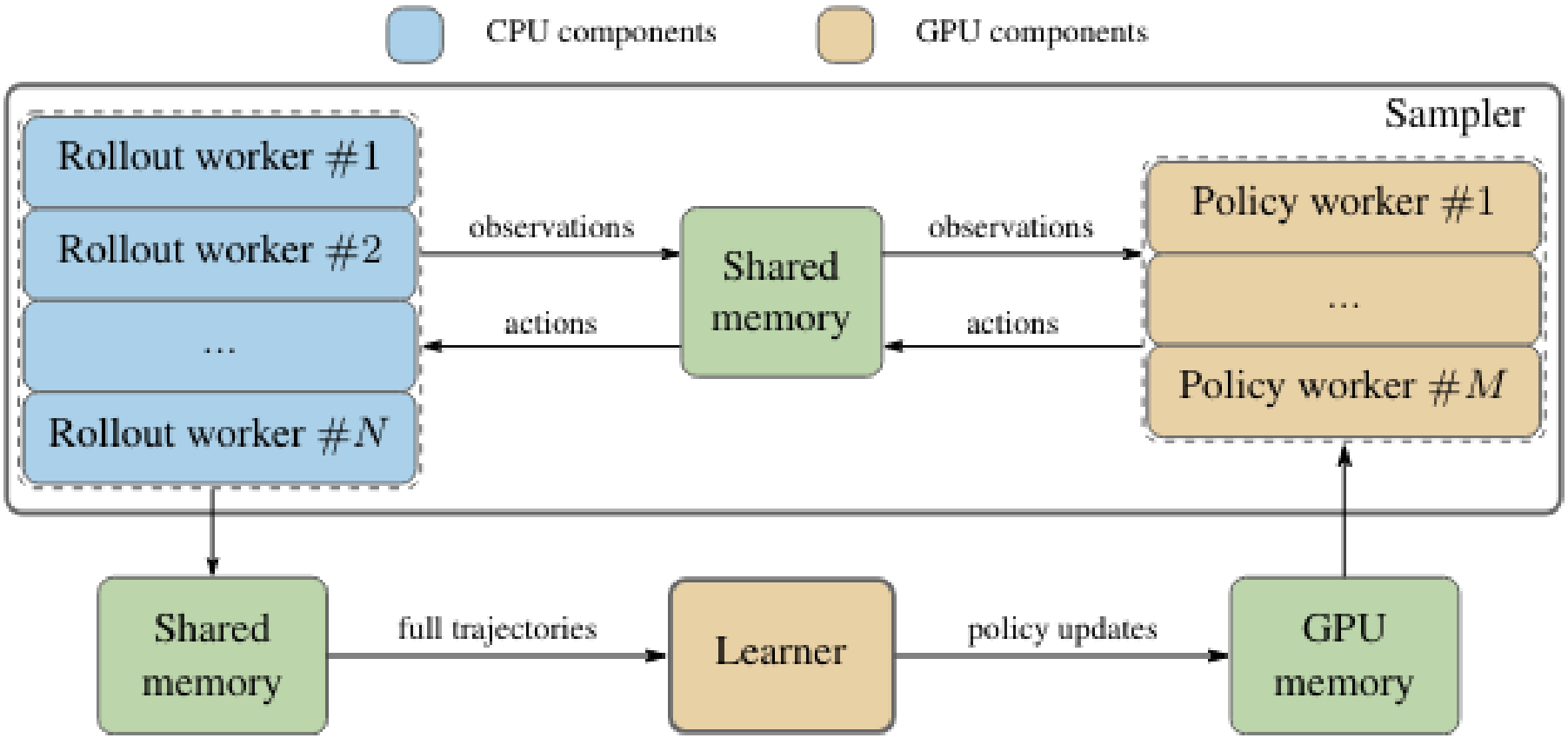



Figure: Overview of the Sample Factory architecture.

# JAT Results for Sequential Decision-Making Tasks [2]

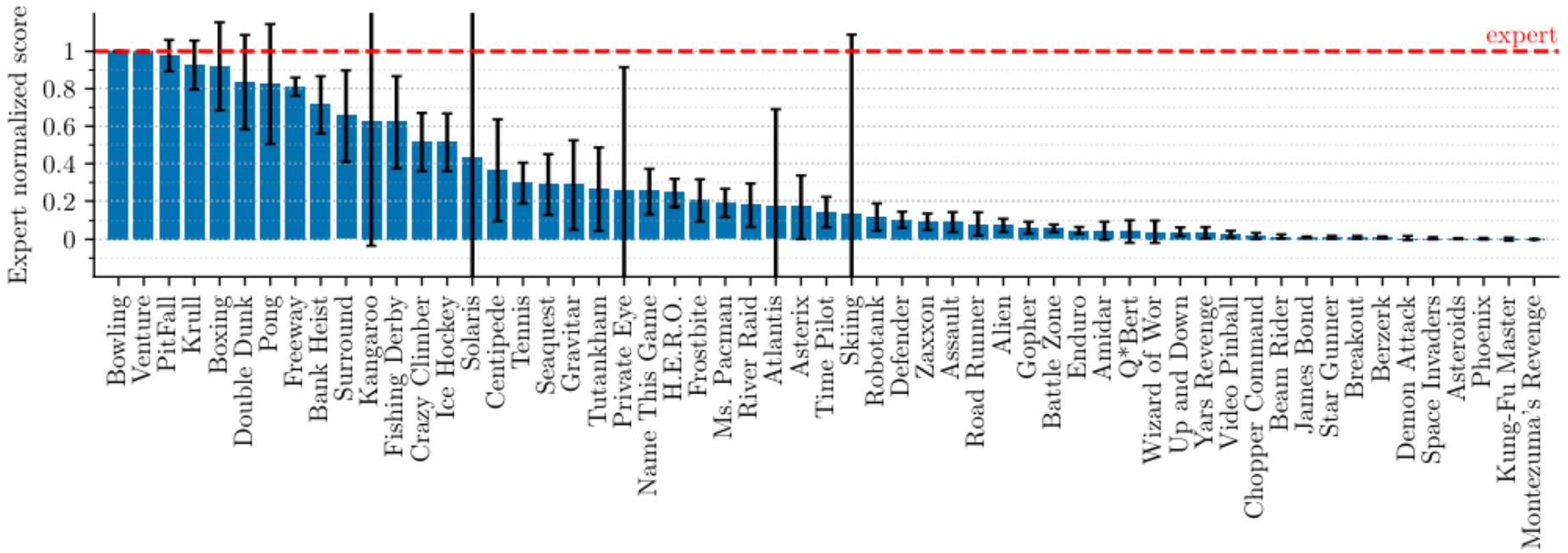



Figure: Expert normalized episodic return for the JAT agent on the Atari 57 benchmark.

# JAT Results for Sequential Decision-Making Tasks [2]

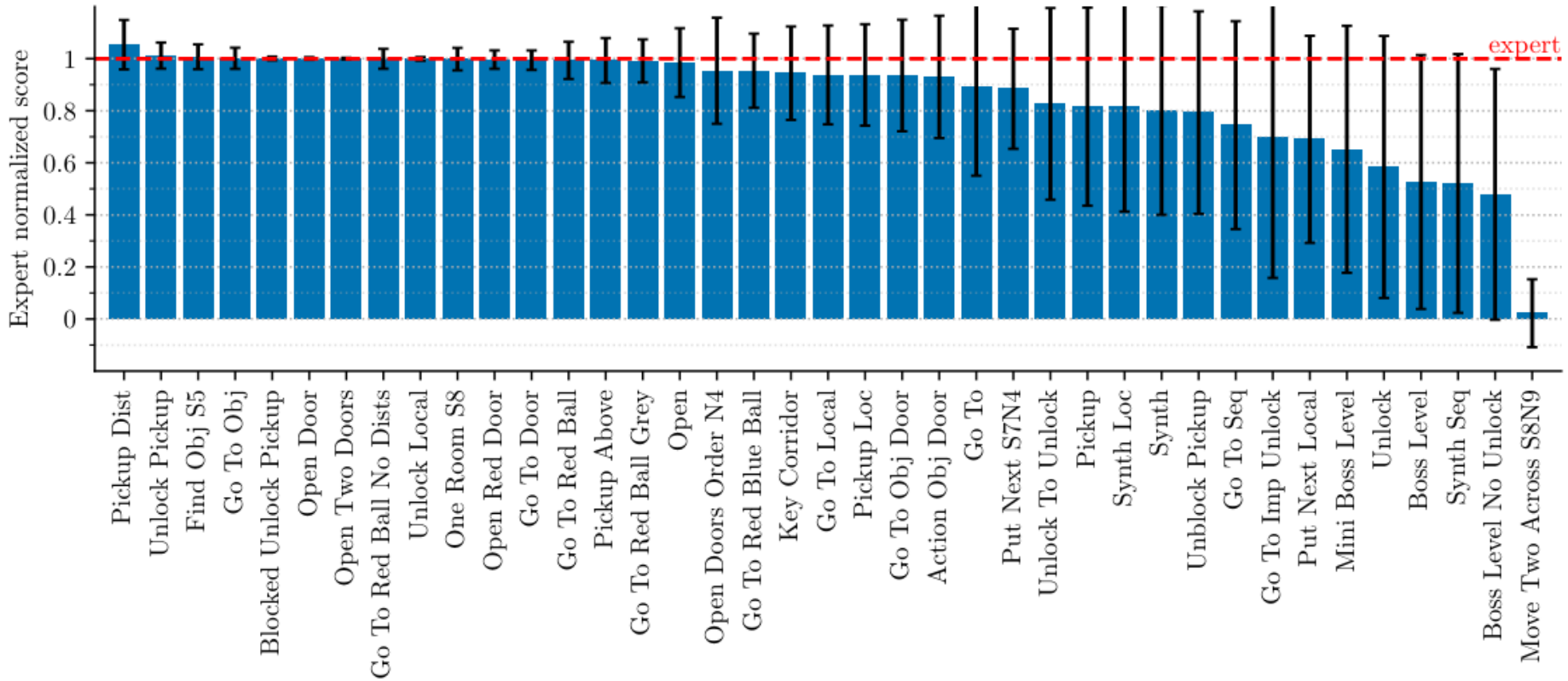



Figure: Expert normalized episodic return for the JAT agent on the BabyAI benchmark.

# JAT Results for Sequential Decision-Making Tasks [2]

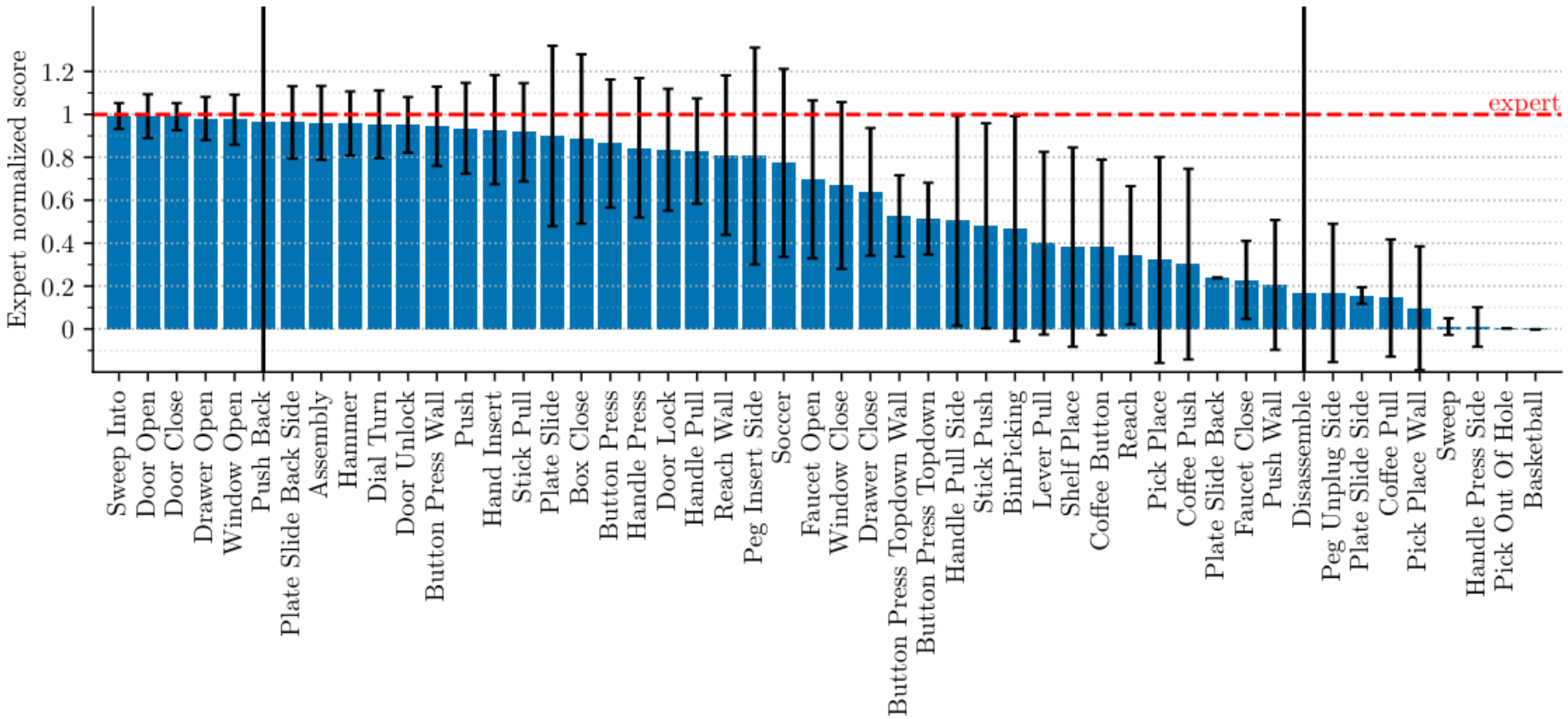



Figure: Expert normalized episodic return for the JAT agent on the Meta-World benchmark.

# JAT Results for Sequential Decision-Making Tasks [2]

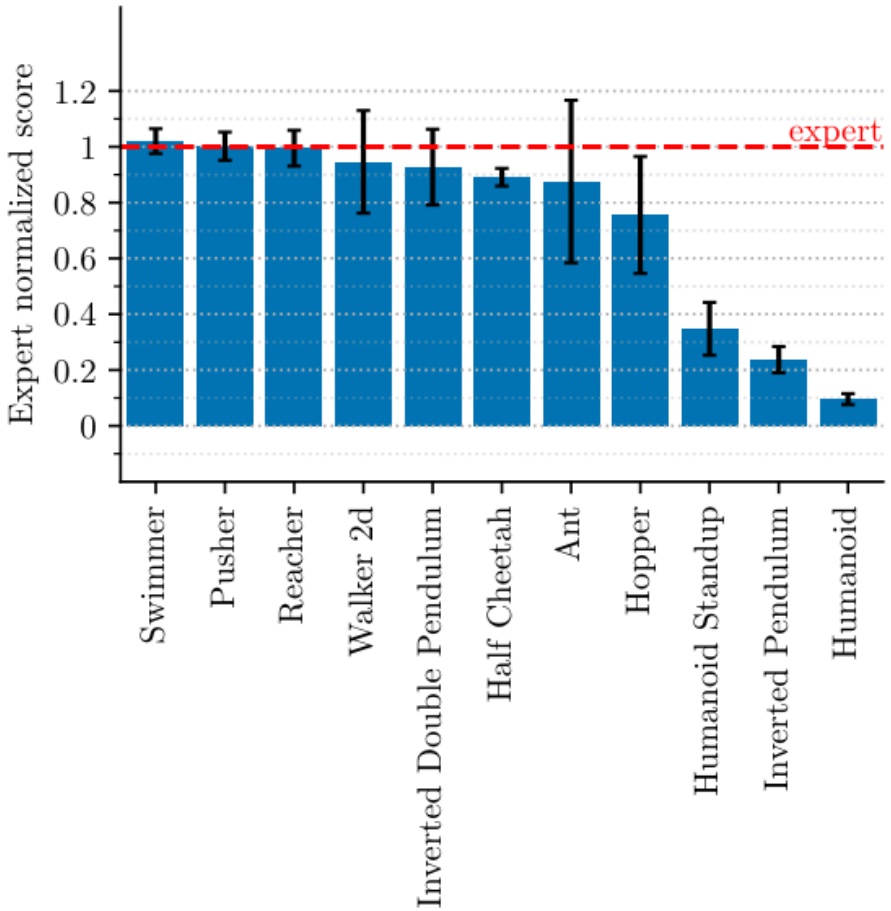



Figure: Expert normalized episodic return for the JAT agent on the MuJoCo benchmark.

# JAT vs Gato [2]

Authors of the paper mainly compare **JAT** model to **Gato** model [4]. Gato is a transformer trained on vision, language, and decision-making tasks without relying on any pre-trained model. Therefore, that model is very similar to JAT but six times larger.

| **Benchmark** | **JAT** | **Gato** |
|---|---|---|
| Atari | 31.1% | 30.9% |
| BabyAI | 86.2% | 93.2% |
| Meta-World | 62.8% | 87% |
| MuJoCo | 73.6% | - |
| DMC | - | 63.6% |

Table: Comparison of average normalized scores for JAT and Gato models.

# RLHF - Reinforcement Learning from Human Feedback [7]

"In this paper, we combine the pre-training advances in natural language processing with human preference learning." [1]

[1] Part of the introduction to the "Fine-Tuning Language Model from Human Preferences" paper

# OpenAI RLHF Timeline

**Fine-Tuning Language Models from Human Preferences**

Daniel M. Ziegler* Nisan Stiennon* Jeffrey Wu Tom B. Brown
Alec Radford Dario Amodei Paul Christiano Geoffrey Irving
OpenAI
{dmz,nisan,jeffwu,tom,alec,damodei,paul,irving}@openai.com

**Learning to summarize from human feedback**

Nisan Stiennon* Long Ouyang* Jeff Wu* Daniel M. Ziegler* Ryan Lowe*
Chelsea Voss* Alec Radford Dario Amodei Paul Christiano*
OpenAI

**Training language models to follow instructions with human feedback**

Long Ouyang* Jeff Wu* Xu Jiang* Diogo Almeida* Carroll L. Wainwright*
Pamela Mishkin* Chong Zhang Sandhini Agarwal Katarina Slama Alex Ray
John Schulman Jacob Hilton Fraser Kelton Luke Miller Maddie Simens
Amanda Askell† Peter Welinder Paul Christiano*†
Jan Leike* Ryan Lowe*
OpenAI

2019

2020

2022

Time

# Pick one

**Learning to summarize from human feedback**

**Nisan Stiennon*** **Long Ouyang*** **Jeff Wu*** **Daniel M. Ziegler*** **Ryan Lowe***

**Chelsea Voss*** **Alec Radford** **Dario Amodei** **Paul Christiano***

OpenAI

Goal of this paper:

- Use a pre-trained large language model and fine-tune it for text summarizing by incorporating human feedback using RLHF

# What is RLHF?

## Reinforcement Learning from Human Feedback

Method to incorporate human feedback into a model by:

1. generating data using the current model,
2. training a reward model,
3. using reinforcement learning to update the main model weights based on the gradients of the advantage estimator and the policy.

Repeat steps 1 through 3 until certain conditions are met.

Soooo.... it is an Advantage Actor-Critic RL.

# How to RLHF?

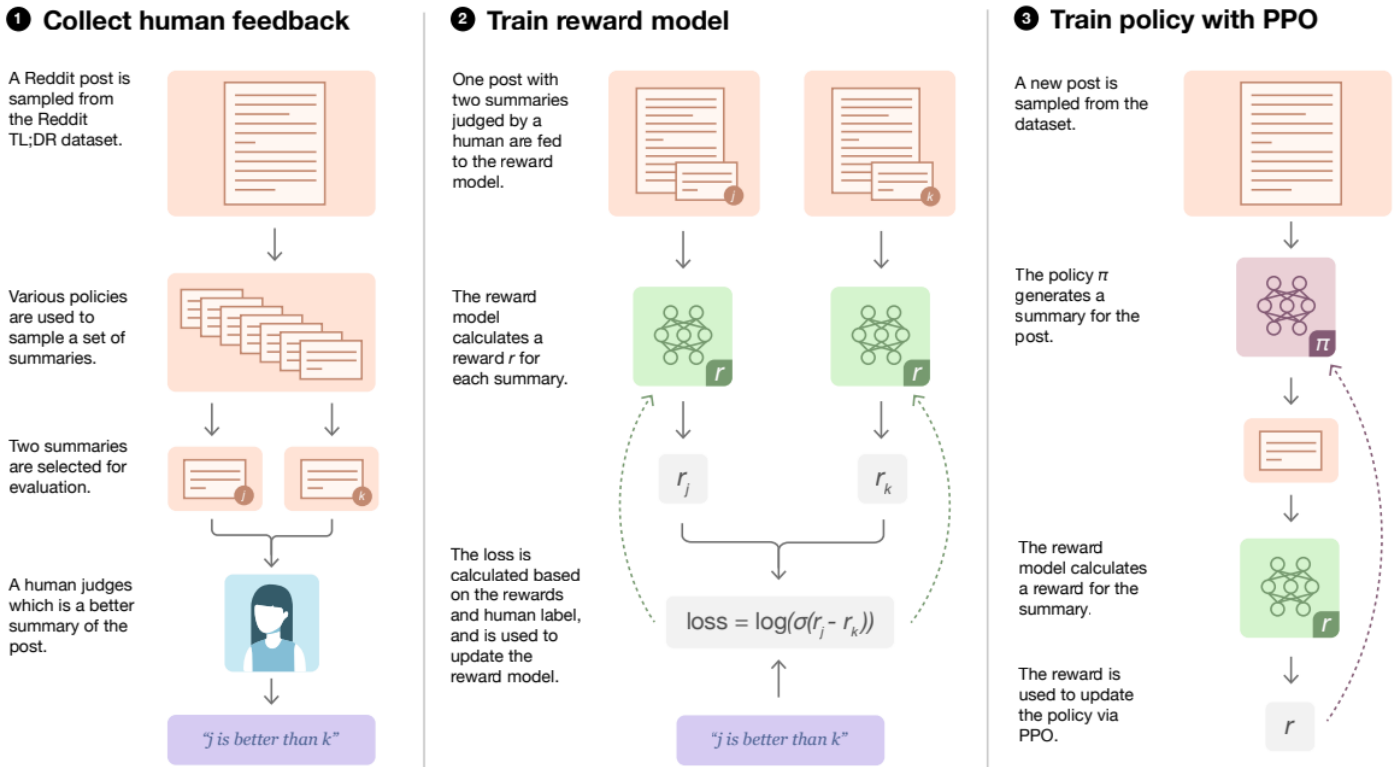
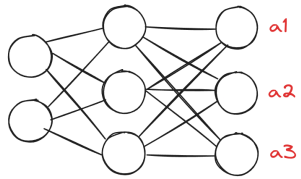



Figure: Diagram of feedback, reward model training, and policy training procedure. Taken from the paper [7]

# How is PPO used here?




LLM
$\pi(a|x)$

Action of choosing a certain word

This is some long text... . Now summarize it: This text is awesome

x

a1 a2 a3 a4

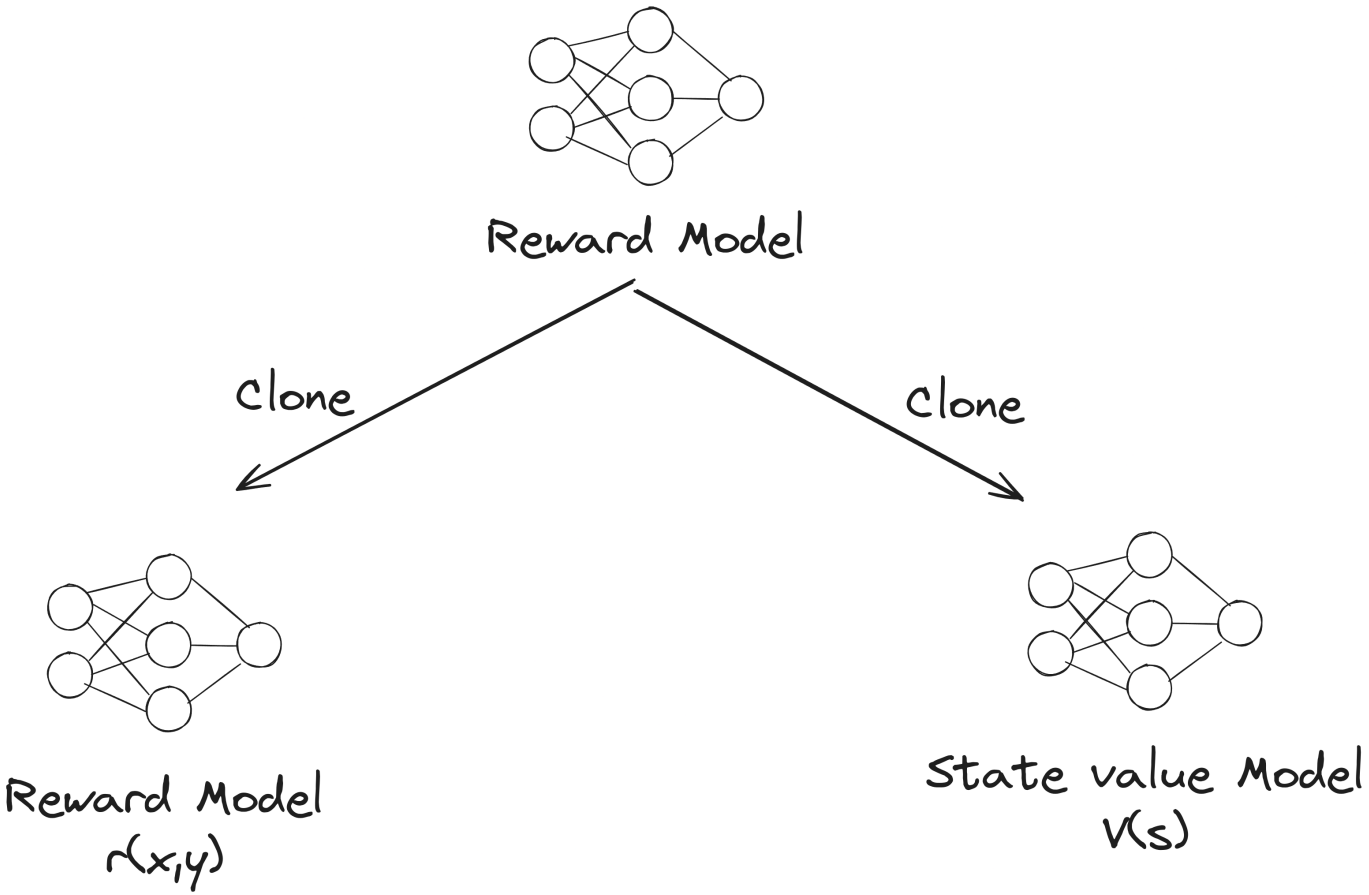
y

s1

s2

s3

# How is PPO used here?



# How is PPO used here?

Authors of the paper modified the reward from the Reward Model to penalize the KL divergence between the learned RL policy and the original pre-trained model:

$$R(x, y) = r_\theta(x, y) - \beta \log \frac{\pi_\phi^{RL}(x|y)}{\pi_\phi^{SFT}(x|y)} \tag{13}$$

# Is RLHF worth-it?

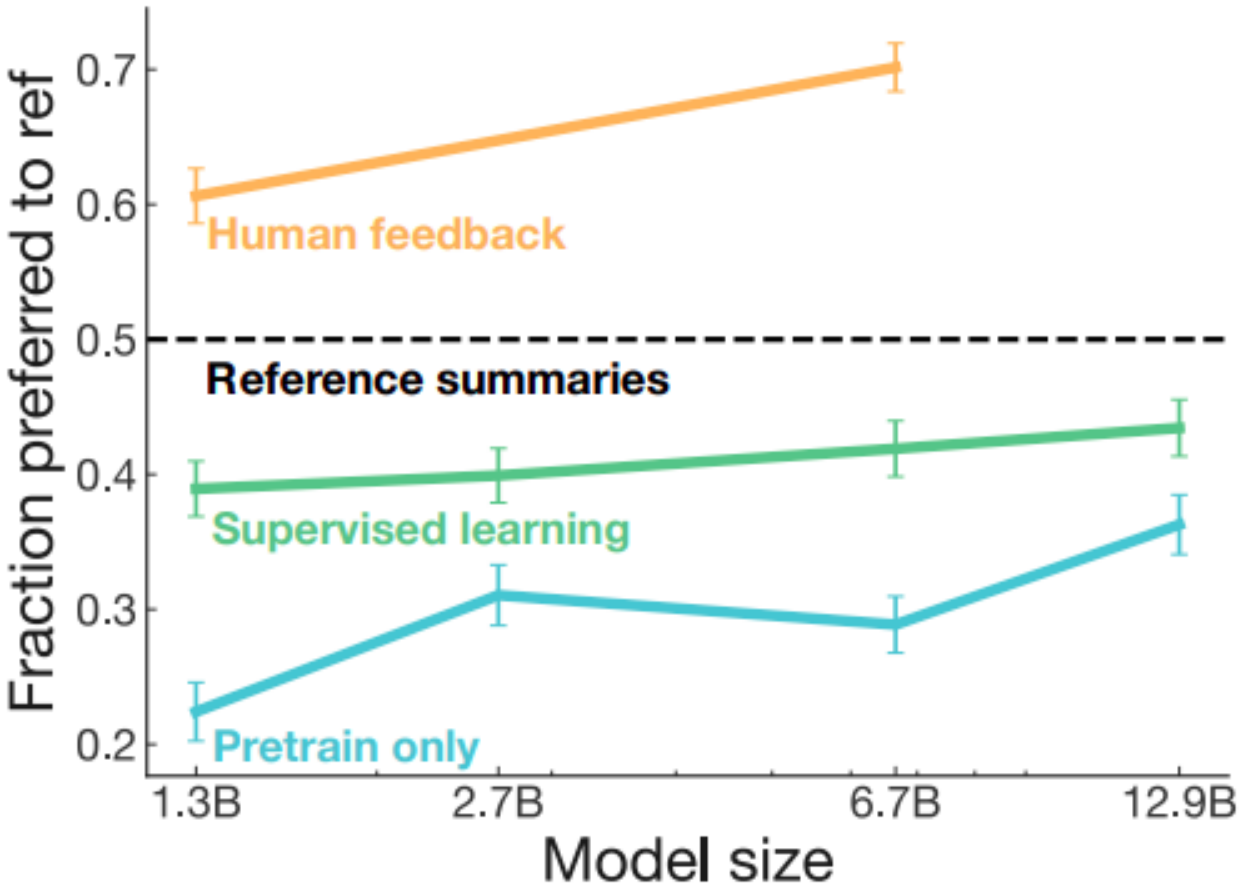



Figure: Fraction of the time humans prefer fine-tuned models' summaries over the human-generated reference summaries on the TL;DR dataset. Taken from the paper [7]

# Autonomous Robots Dexterity

## Core Challenge

Achieving dexterous in-hand manipulation using a robotic hand without human demonstrations.

- **Complexity:** High dexterity requirements simulating human hand capabilities.
- **Environment:** Learning entirely through simulations.
- **Goal:** Transfer simulated learning to real-world robot applications.

## Innovative Simulation Techniques

Extensive use of randomized simulations to train robust control policies.

- **Domain Randomization:** Introduce variability in physical and visual parameters within the simulation.
- **Scalability:** Employ distributed reinforcement learning to handle complex and variable tasks.
- **Real-world Transfer:** Ensure learned behaviors translate effectively from simulation to real robot execution.

## Recurrent Neural Network (RNN) with LSTM

Utilizes LSTM to maintain internal state information crucial for sequence prediction in manipulation tasks.

- **Network Inputs:** Receives sensory data from simulated environment.
- **PPO Algorithm:** Employs Proximal Policy Optimization to efficiently train the policy and value networks.
- **Learning Dynamics:** Asynchronous updates facilitate learning complex manipulative actions within varied simulation scenarios.

# Autonomous Robots Dexterity

## Achievements

Significant mastery of dexterous manipulation in simulated and real environments.

- **Simulation Success:** High success rates in manipulating a variety of objects within simulation settings.
- **Real-World Application:** Effective transfer of learned behaviors to the physical robot, despite complexities.
- **Behavioral Emergence:** Natural emergence of human-like grasps and manipulative strategies.

# Honorable Mentions

List of other papers that also make use of the PPO:

- A graph placement methodology for fast chip design
  `https://www.nature.com/articles/s41586-021-03544-w`
- Efficient Reinforcement Learning for Autonomous Driving with Parameterized Skills and Priors
  `https://arxiv.org/pdf/2305.04412`
- MnasFPN : Learning Latency-aware Pyramid Architecture for Object Detection on Mobile Devices
  `https://arxiv.org/pdf/1912.01106v2`
- SpineNet: Learning Scale-Permuted Backbone for Recognition and Localization
  `https://arxiv.org/pdf/1912.05027v3`

# Table of Contents


1. Introduction
2. Related work
   - Q-learning
   - Vanilla gradient method
   - Trust Region Policy Optimization (TRPO)
3. How does PPO work?
4. PPO Results in the Paper
5. PPO in the literature
   - Jack of all Trades
   - RLHF - Reinforcement Learning from Human Feedback
   - Autonomous Robots Dexterity
   - Honorable Mentions
6. Summary
7. Bibliography


# Summary

**Proximal Policy Optimization** is stable, reliable and simple to implement. It allows for easy parallelism for large-scale projects. PPO has proven to be applicable in a wide range of use cases, e.g. Atari games, robotic tasks or text summarization, with comparable or better performance compared to other RL methods.

# Table of Contents


1. Introduction
2. Related work
   - Q-learning
   - Vanilla gradient method
   - Trust Region Policy Optimization (TRPO)
3. How does PPO work?
4. PPO Results in the Paper
5. PPO in the literature
   - Jack of all Trades
   - RLHF - Reinforcement Learning from Human Feedback
   - Autonomous Robots Dexterity
   - Honorable Mentions
6. Summary
7. Bibliography


# Bibliography I


[1] D. Bick. *mAI: Towards More Ethical and Inclusive Artificial Intelligence Systems*. https://fse.studenttheses.ub.rug.nl/25709/1/mAI_2021_BickD.pdf. Bachelor's Thesis. 2021.

[2] Quentin Gallouédec et al. *Jack of All Trades, Master of Some, a Multi-Purpose Transformer Agent*. 2024. arXiv: 2402.09844 [cs.AI].

[3] Aleksei Petrenko et al. "Sample Factory: Egocentric 3D Control from Pixels at 100000 FPS with Asynchronous Reinforcement Learning". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, pp. 7652–7662. URL: https://proceedings.mlr.press/v119/petrenko20a.html.


# Bibliography II


[4] Scott Reed et al. "A Generalist Agent". In: *Transactions on Machine Learning Research* (2022). Featured Certification, Outstanding Certification. ISSN: 2835-8856. URL: https://openreview.net/forum?id=1ikK0kHjvj.

[5] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG].

[6] John Schulman et al. *Trust Region Policy Optimization*. 2015. DOI: 10.48550/ARXIV.1502.05477. URL: https://arxiv.org/abs/1502.05477.

[7] Nisan Stiennon et al. *Learning to summarize from human feedback*. 2020. DOI: 10.48550/ARXIV.2009.01325. URL: https://arxiv.org/abs/2009.01325.


Thank You for Your Attention!