Extended Abstract

Motivation Human motion imitation is a challenging task in robotics and character animation, because it requires the development of sophisticated control systems that can accurately reproduce complex, dynamic movements. The ability to transfer human motion data to robotic systems can have significant implications on physical robot and animation. In the past, there have been successful attempts on training policies that can reproduce one reference motion (Parisotto et al. (2015)). However, training a policy that can perform multiple motions is challenging due to catastrophic forgetting, need for large model architectures (e.g., transformers, diffusion models), datasets, and significant investments in compute resources. Therefore, in this project, we focus on developing a unified lighweight policy network that can successfully replicate multiple complex human motions.

Method Our end-to-end pipeline comprises of three stages. In the first stage, we trained a single-motion control policy network to reproduce motions using DeepMimic (Parisotto et al. (2015)). For the purposes of this project, we picked six different motions – walk, skip to walk, crouch, crouch to lie, swing, and crawl motions. In the second stage, we "distilled" the 6 expert motions into one multi-action policy using imitation learning. We experimented with expert data relabeling, and using dataset aggregation to offset compunding errors. In all of the variations, the agent is updated by the negative log likelihood. In the last stage, we fine-tuned the unified model further using PPO in the target environment. This allows the policy to improve performance and robustness in reproducing multiple motions cohesively.

Implementation We use MLP as our default model architecture for each of expert, actor, and critic networks at different stages of the training. This results in a compact, lightweight model that can be readily used in downstream applications. We performed all simulation using Isaac Gym (Makoviychuk et al. (2021)), each with 4096 parallel environments. The human motions are randomly picked from the AMASS dataset (Mahmood et al. (2019)), ensuring sufficient variability for the challenging task. We evaluate the performance of our models using two main metrics: tracking success rate (↑), and global translation error (↓).

Results In our first stage, we find that all the expert networks were able to successfully learn their respective motions, although some required significantly more training than others. The walking motion converged within 400 epochs, while the remaining motions typically required between 4k-5k epochs. In the second stage, we trained three different variations - **(A)** pure behavior cloning agent, **(B)** 25% BC + 75% expert data relabelings (DAgger 75%), and **(C)** 75% BC + 25% expert relabelings (DAgger 25%). We found that using **(B)** 25% BC + 75% DAgger achieved the best performance successfully replicating 5/6 motions (success rate = 0.83). In contrast, **(C)** 75% BC + 25% DAgger was the worst performing, succeeding on only 3/6 motions (success rate = 0.5). At the last stage, we observed that PPO improved the performances of all models from Stage 2, achieving an impressive success rate of 1. Interestingly, even when the model was initialized from scratch, direct PPO finetuning with Stage 1 experts resulted in a success rate of 1.

Discussion This project provides valuable insights into imitation learning, reinforcement learning, and multi-task learning. While DAgger improved performance, it is also worth highlighting the risks of using neural network experts, which can behave unpredictably under distribution shifts. Moreover, our experiments have been conducted on a very small scale (just 6 motions). Due to stochastic nature of the training process, more experiments are needed to verify the trends we observed, for e.g., direct PPO finetuning (w/o Stage 2 training) achieves a success rate of 1 easily, contradicting the two-stage training paradigms (supervised pre-training + finetuning) prevalent in the research community. We are unsure if these trends will hold when scaled to thousands/millions of human motions.

Conclusion In this project, we explored the imitation learning, and PPO finetuning paradigms to learn a single expert policy capable of replicating six different human motions. Based on the experiments we conducted, we find that imitation learning is somewhat successful in replicating some of the motions (success rate $\sim$ 0.5-1). On the other hand, PPO-finetuned models are able to achieve this objective successfully (success rate of 1), even when initialized from scratch. In the future, we think it would be a good idea to confirm some of the trends we observed by collecting more empirical data.

Learning from Experts: Three Stage Training for Multi-Action Physics-Based Control

Andi Xu
Department of Computer Science
Stanford University
andixu@stanford.edu

Eric Chen
Department of Computer Science
Stanford University
erchen22@stanford.edu

Silky Singh
Department of Computer Science
Stanford University
silsingh@stanford.edu

Abstract

In this project, we focus on training a single, unified policy network that can successfully perform multiple motions similar to a human. Existing literature suggests that imitating *multiple* complex human motions is difficult to achieve. To this end, we propose a three-stage training pipeline that accomplishes the objective for a total of six human motions – *walk, crouch, skip to walk, crouch to lie, swing, and crawl*. These motions are different from each other, possibly covering entirely different distributions of the action space. We train an expert network for each motion separately, and distill these models into a single policy network using imitation learning objective. Focussing on robustness, sample efficiency and generalization objectives, we further finetune our policy network using Proximal Policy Optimization (PPO). Our results suggest that PPO improves the success rate in all different versions of the model we trained. Interestingly, if we skip Stage 2 to perform direct PPO, the model still achieves a success rate of 1, i.e., it can successfully replicate all six motions simply learning from the experts.

1 Introduction

Human motion imitation is a challenging task in robotics and character animation, because it requires the development of sophisticated control systems that can accurately reproduce complex, dynamic movements. The ability to transfer human motion data to robotic systems can have significant implications on physical robot and animation. In the past, there have been successful attempts on training policies that can reproduce one reference motion Parisotto et al. (2015). However, training a policy that can track multiple motion tracking is challenging. When training a policy on motions that look different from each other, it is easy for the policy to forget previously learned motions when learning new ones. Moreover, previous work requires massive computing resources, using deep architecture on large datasets Tessler et al. (2024); Luo et al. (2023); Truong et al. (2024). How to solve the issue with limited compute is also a challenge.

In our project, we aim to train a multi-motion physics-based controller that is more data-efficient and uses small, compact models. We experimented with a 3 stage training pipeline: first we train single motion models similar to DeepMimic; second we use behavior cloning (BC) and data aggregation (DAgger) to distill the first stage models into a single policy, and finally, we fine-tune the policy online with PPO to improve its robustness. We used a part of the AMASS dataset for our experiments, and ran simulation on IsaacGym Makoviychuk et al. (2021). Our evaluation focuses on two numerical

metrics that capture different aspects of tracking performance: Global Translation Error (gt_err), which quantifies the spatial accuracy of body segment positioning, and Tracking Success Rate, which provides a binary assessment of whether motion sequences are executed within acceptable error bounds. These metrics together provide a comprehensive view of both the precision and reliability of the learned control policies. besides the two metrics, we visualized our trained policy to understand our results qualitatively. To summarize, this project focuses on the following research questions:

1. Would imitation learning be successful in training a single MLP policy network to replicate complex human motions?

2. How effective would Proximal Policy Optimization (PPO) be in finetuning a unified policy network?

3. Are small and lightweight models (e.g., a two-layer MLP) complex enough to distill knowledge from multiple experts?

2 Related Work

One of the early breakthroughs in generating physics-based character skills is Deepmimic Parisotto et al. (2015). Deepmimic is a method to imitate skills by training on reference motion clips. To mimic motion, a policy model is trained PPO style, where the policy model takes in states (and optionally depth map and goal states) and outputs target joint orientations. The reward for imitation is calculated through several comparison metrics between the generated positions and reference joint orientations and relative positions. Deepmimic can also incorporate a goal state and a goal-specific reward so that the policy can learn to perform actions like throwing a ball to a specific target location. Finally, the authors also designed several ways to learn different motion types and from different motion clips. One such innovation, the skills selector policy, allows users to choose from multiple actions by providing the corresponding one-hot vector. In our project, we also seek to combine multiple policies into one, but we hope to choose actions from the policy through natural language.

Recently, MaskedMimic Tessler et al. (2024) managed to use a single policy to reproduce different tasks. It introduces a single physics-based humanoid controller that treats character control as a motion-inpainting problem: the policy receives a partially "masked" trajectory — e.g., sparse key-frames, object targets, or text instruction — and "recover" the missing poses while being physically natural. The authors first trained a general policy that learns to mimic full-body trajectories of a character, then train a distilled model with randomly masked inputs of different modalities. Therefore, the second distilled network accepts many control interfaces without per-task reward engineering, achieving state-of-the-art versatility for tracking, path following, object interaction, and text-conditioned motions in a unified framework. However, this work requires multiple training steps and still requires goal-engineering for complex tasks.

PDP (**?**) leverages expert policies to create a large, diverse, and stochastic dataset, which it then uses to train a diffusion-based policy network that can accomplish a variety of downstream tasks. The key insight is that expert RL policies can provide corrective actions from sub-optimal states. By changing the sampling strategy from commonly used [clean state : clean action], or [noisy state : noisy action] pairs to [noisy state : clean action] pairs, it effectively makes an error band around the trajectories to learn more robust policies. However, the method is T times slower compared to non-diffusion based policy networks, and suffers with tradeoff in diversity versus accuracy of the actions. This work could serve as a good starting point to create datasets for variety of tasks using the expert policies already trained for PDP. Moreover, it provides a working architecture for cross-attention mechanism that allows us to condition on text inputs from users.

3 Method

We aim to train a unified policy capable of reproducing diverse motion skills using multiple motion clips. Our approach follows a three-stage pipeline (refer Fig. 1).

3.1 Expert Policy Training with DeepMimic

In the first stage, for each of the six motion clips that we picked (walk, skip to walk, crouch, crouch to lie, swing, and crawl), we trained one single-motion control policy to reproduce that motion, using DeepMimic Parisotto et al. (2015).

At each timestep, besides the goal motion g_t, observation s_t is the current state of the character, consisting of the 3D body pose and velocity, all calculated based on the character's local coordinate frame. Different from DeepMimic, we took a further step from MaskedMimic that in s_t, we also observe the next K target poses from the reference motion. All joint data is also canonicalized relative to the current root and current respective joint. For scene observation, the controller also receives a height map of the terrain under the character's root. But in our project, we did not consider special terrain besides flat ground. Action a_t is represented by proportional derivative (PD) control to joints. Each policy π is represented by a neural network that maps a given state s and goal g to a distribution over action $\pi(a|s, g)$. The action distribution $\pi(a_t|s_t, g_t)$ is represented using a multi-dimensional Gaussian with a fixed diagonal covariance matrix. The actor network is a two-layer fully-connected network with 1024, and 512 units each. We used ReLU activations for hidden units. The critic network is a similar fully-connected network with the output layer consisting of a single linear unit.

We leveraged the Deepmimic implementation from MaskedMimic Tessler et al. (2024) codebase. The only notable difference is that the MaskedMimic version provides future k reference motion information, as mentioned above. The trained single-motion control policies will be used as expert policies for the second stage's imitation learning.

3.2 Unified Policy Initialization via Imitation Learning

In the second stage, we "distilled" the 6 expert motions into one multi-action policy using imitation learning. At each iteration, the policy performs a rollout when provided with future poses of a reference motion. We experimented with three variations here. First, we only use data from expert data, doing simple behavior cloning. Second, we apply DAgger by mixing on-policy rollouts with expert corrections iteratively. We let the corresponding expert relabels 75% of the actions (75% relabelled data, 25% expert data). In the third variation, we still use DAgger, but 25% relabelled data, 75% expert data. In all of the three variations, the agent is updated by the negative log likelihood.

3.3 Fine-tuning with PPO

In the last stage, we fine-tuned the trained unified model from the second stage using PPO in the target environment. This allows the policy to further improve performance and robustness in reproducing multiple motions cohesively. Because we aim to train a computation-efficient model, we choose MLP for both actor's and critic's architecture.

We used similar reward r_t to the one MaskedMimic used in their first stage controller. The goal is to let the character minimize the difference between the state of the simulated character and the target motion so that it can reproduce reference motion:

$$r_t = w^{gp} \cdot r_t^{gp} + w^{gr} \cdot r_t^{gr} + w^{rh} \cdot r_t^{rh} + w^{jv} \cdot r_t^{jv} + w^{jav} \cdot r_t^{jav} + w^{eg} \cdot r_t^{eg},$$

where $r_t\{\cdot\}$ is different reward score and and $w\{\cdot\}$ is respective weight. Different terms encourages the character to imitate the reference motion's global joint positions (gp), global joint rotations (gr), root height (rh), joint velocities (jv), joint angular velocities (jav), and an energy penalty (eg) to encourage smoother and less jittery motions Tessler et al. (2024).

4 Experimental Setup

We performed all simulation using Isaac Gym Makoviychuk et al. (2021), each with 4960 parallel environments. All-stage controllers operate at 30 Hz. Detailed hyperparameter settings are available in the supplementary material.

4.1 Dataset

We used the AMASS dataset for our project, a large database of human motion unifying different optical marker-based motion capture datasets by representing them within a common framework and

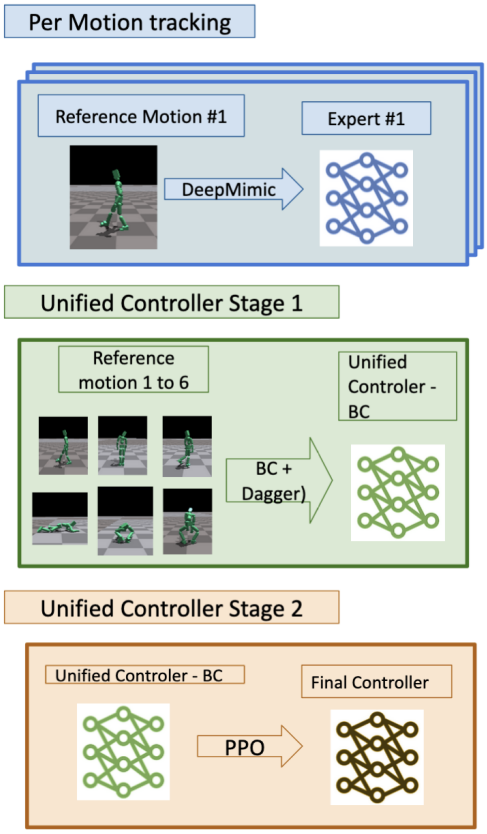


Figure 1: Method Overview. Our end-to-end method is comprised of three stages. **Stage 1:** Expert training for each motion, **Stage 2:** A unified controller trained using Behavior Cloning and DAgger to mimic the experts from Stage 1, and **Stage 3:** The unified controller from Stage 2 finetuned using PPO

parameterization Mahmood et al. (2019). Previous work has observed some motions in the AMASS dataset contains artifacts Luo et al. (2023); Juravsky et al. (2024) that make them unsuitable for being used as part of the training data. Therefore, we used the same filtering approach as PHC Luo et al. (2023) to first obtain a filtered dataset by removing data that is noisy or involve interactions of human objects. After that, considering our limited computing resources, we selected 6 motion clips from the sub-dataset ACCAD Advanced Computing Center for the Arts and Design ([n. d.]) to used as our training data: walking, skip to walk, crouch, crouch to lie, crawl, and swing. These representative motion clips span diverse locomotion and manipulation styles, therefore can represent the challenging aspect of our goal. Each clip ranges from about 5 seconds (150 frames at 30 Hz). All motions were retargeted to the smpl humanoid skeleton Loper et al. (2015), normalized for height and orientation. The reference state at each timestep consists of joint angles, joint angular velocities, and root position and orientation.

4.2 Evaluation Metrics

We analyze the performance of our final unified model using two quantitative metrics, a success rate metric and an error rate metric.

4.2.1 Tracking Success Rate

The unified controller is required to closely follow the joint angles of the joint motions. The `tracking_success_rate` is a binary success metric that determines whether an agent successfully tracks a motion sequence without significant deviations. It is computed as follows. First, we decide a failure threshold. A motion is considered failed if the maximum global translation error

(gt_err_max) exceeds 0.5 meters at any point during the sequence execution. Then each motion sequence is classified as either successful (tracking error remains below threshold) or failed. The overall success rate is calculated as the complement of the failure rate: $\texttt{tracking\_success\_rat} = 1.0 - \texttt{tracking\_failures.mean()}$. A tracking success rate of 1 indicates that all motion sequences were executed within acceptable error bounds, while lower values indicate the proportion of motions that were successfully tracked.

4.2.2 Global Translation Error(gt_{err})

gt_{err} quantifies the positional accuracy of the humanoid's body segments compared to reference motion data. The current body positions gt are first adjusted relative to the reference data by removing terrain height offsets and respawn position offsets to ensure fair comparison with the ground-truth motion data ref_{gt}. Then for each body segment, at each timestamp, the Euclidean distance between the predicted and reference positions is computed. The final gt_{err} is obtained by averaging the per-joint errors across all J body segments: $gt_{err} = \frac{1}{J}\Sigma_j||ref_{t,j} - p_{t,j}||$.

This metric provides a comprehensive measure of how accurately the agent reproduces the spatial positioning of the reference motion, with lower values indicating better tracking performance. The error is measured in meters.

5 Results

Experiments	Tracking Success Rate (↑)	Eval Ground Truth Error (↓)
Stage 1: Training Experts		
Walk	1.00	0.0538
Skip to Stand	1.00	0.0488
Crounch	1.00	0.0500
Crouch to Lie	1.00	0.3830
Crawl	1.00	0.0590
Swing	1.00	0.0272
Stage 2: Unified Controller (Behavior Cloning + DAgger)		
BC w/o DAgger	0.67	0.6563
BC 75%, DAgger 25%	0.50	0.6285
BC 25%, DAgger 75%	0.83	0.0943
Stage 3: PPO Fine-Tuning		
BC w/o DAgger	0.83	0.2123
BC 75%, DAgger 25%	0.83	0.5944
BC 25%, DAgger 75%	1.00	0.0649
Baseline: Direct PPO w/o BC	1.00	0.0502

Table 1: Best Tracking Success Rate and the corresponding gt_err for all runs.

5.1 First Stage: Expert Model

We trained a separate DeepMimic-style model for each motion clip to serve as an expert policy. The success rates and corresponding gt_err values are shown in Table 1. All expert models were able to successfully learn their respective motions, although some required significantly more training than others. For example, the walking motion converged within 400 epochs, while the remaining motions typically required between 4,000 and 5,000 epochs.

The final ground-truth errors were low across all experts, with most achieving a gt_err of approximately 0.05 meters. Qualitatively, the motions generated by these expert policies closely resemble natural human movement. Across repeated rollouts, the experts consistently reproduced their target motions when initialized from the correct starting pose. However, the policies were not robust to arbitrary starting conditions—for instance, if the agent began in a falling posture, it was unable to recover and complete the intended motion.

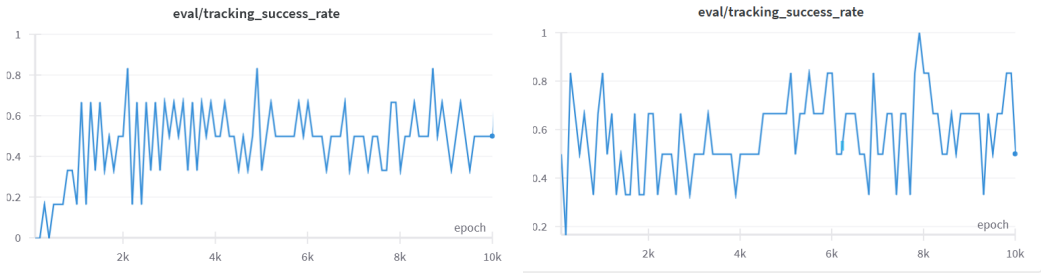


Figure 2: **Left**: BC 25% + DAgger 75% **Right**: + PPO Finetuning Success Rate vs Epoch

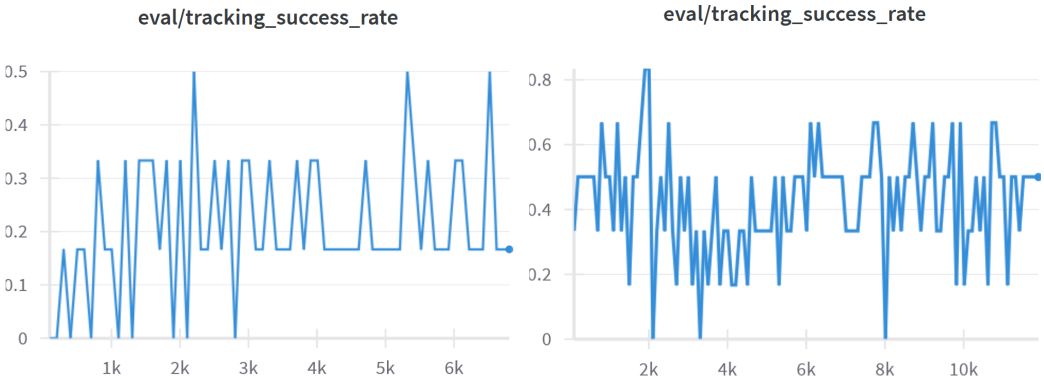


Figure 3: **Left**: BC 75% + DAgger 25% **Right**: + PPO Finetuning Success Rate vs Epoch

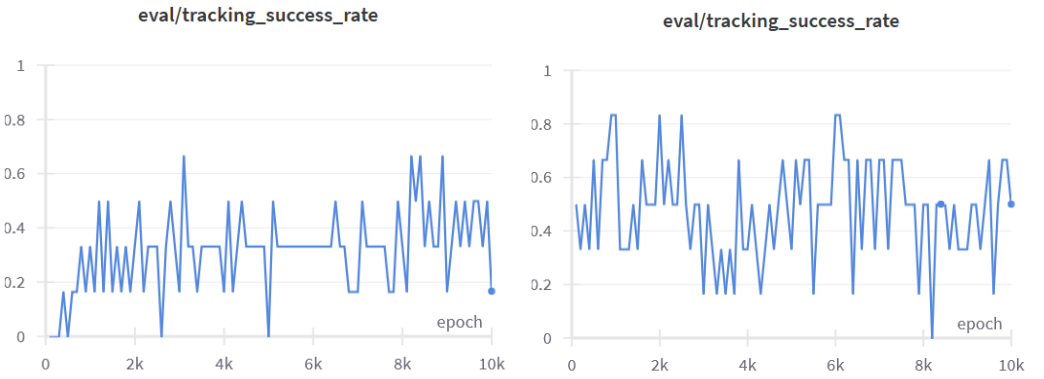


Figure 4: **Left**: BC w/o DAgger **Right**: + PPO Finetuning Success Rate vs Epoch

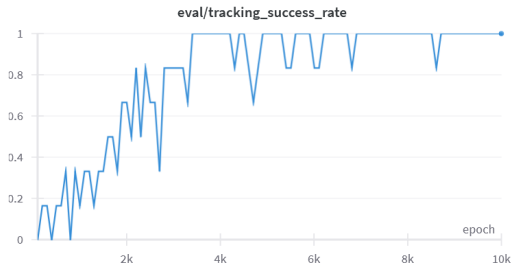


Figure 5: Direct PPO (Baseline) on 6 Motions Success Rate vs Epoch

5.2 Second Stage: Imitation Learning Model

We used the expert models from Stage 1 to train our first set of unified motion models. We experimented with three configurations:

- Pure Behavior Cloning (BC)
- 25% BC + 75% expert relabelings (DAgger 75%)
- 75% BC + 25% expert relabelings (DAgger 25%)

All approaches were trained for approximately 10,000 total epochs, with the first 1,000 epochs dedicated exclusively to behavior cloning using expert rollouts.

The success rates and ground-truth errors are reported in Table 1. The configuration with 25% BC and 75% DAgger achieved the best performance, reaching a success rate of 0.8333 (successfully executing 5 out of 6 motions). In contrast, the 75% BC + 25% DAgger performed the worst, with a success rate of 0.5 (succeeding on only 3 out of 6 motions).

The ground-truth errors follow a similar trend: both the pure BC and the DAgger 25% models resulted in high average errors around 0.6 meters, while the DAgger 75% model achieved a significantly lower average error of 0.09 meters. **These results suggest that a lightweight two-layer MLP can successfully reproduce multiple motion types using a single unified model, and further validate imitation learning as an effective "distillation" strategy for combining expert behaviors.**

Qualitative examples of agent behavior are shown in Figures 6 to 9. Even when successful, the agent's motions do not always appear natural—the trajectories are roughly correct, but the motion style lacks realism. This is expected, as the model is not explicitly trained to match the style of the expert policies (e.g., matching joint velocities), which would require a more detailed reward function. The negative log-likelihood loss used here is a relatively coarse performance signal.

During inference, the policy is not entirely reliable. Certain motions, especially crawling, may require multiple attempts to succeed. This instability is also reflected during training, where the success rate fluctuates significantly.

Our results also indicate that higher levels of DAgger generally improve performance, supporting the intuition that DAgger helps correct compounding errors. This confirms that DAgger can serve as a viable distillation strategy for aggregating individually trained expert policies, consistent with findings from prior multi-motion learning work Juravsky et al. (2024).

One concern we had was that the experts—being narrowly trained and brittle to out-of-distribution states—might produce incorrect corrective labels. However, this does not appear to be a significant issue in our case. After the initial 1,000-epoch BC warm-up, both DAgger variants showed improvement in success rates, suggesting that the warmed up model has learned enough to perform within the expert's state distribution, and therefore able to benefit from correct corrective feedback.

Interestingly, the 75% BC + 25% DAgger configuration performed worse than both more extreme configurations (pure BC and DAgger 75%). This is unexpected and suggests a non-linear interaction between expert data and corrective relabeling. Additional data and experimentation would be needed to confirm this trend, especially given the stochastic nature of the training process.

5.3 Final model

We applied PPO fine-tuning to each of the unified models from Stage 2. As shown in Table 1, fine-tuning generally led to higher tracking success rates and lower ground-truth errors compared to their respective pre-finetuned versions. However, the only configuration that successfully learned all six motions was the BC 25% + DAgger 75% model.

Interestingly, our baseline method—training a unified policy from scratch using only PPO—also achieved a perfect success rate of 1.0. This suggests that PPO fine-tuning may be essential for learning more challenging motions, even when starting from a reasonably competent policy.

When comparing training efficiency, the success rate versus training epoch curves (Figures 2 to 5) reveal that the direct PPO baseline learns all six motions in just 5,000 epochs—faster than the imitation learning-based approaches. This is somewhat surprising, as PPO learns purely via trial and error, without access to expert demonstrations.

Figure 6: **Left:** Full BC walking (fails) **Right:** Full BC followed by PPO Finetuning walking (success)

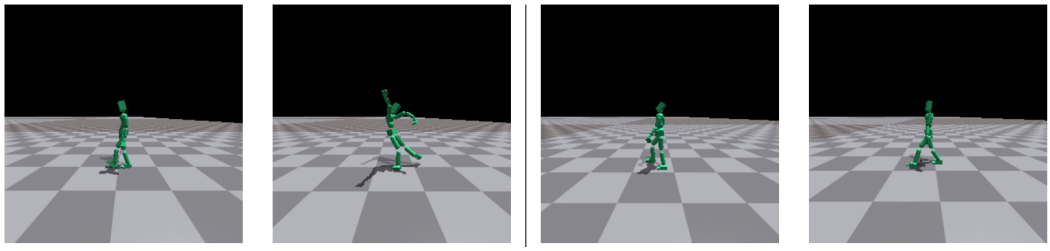


Figure 7: **Left:** Full BC skipping (fails) **Right:** Full BC followed by PPO Finetuning skipping (fails)

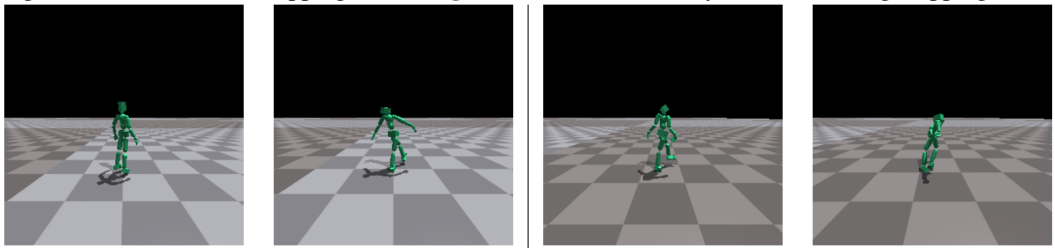


Figure 8: **Left:** 75% BC + 25% DAgger Crouch to Lie (fails) **Right:** 75% BC + 25% DAgger followed by PPO Finetuning Crouch to Lie (succeeds)

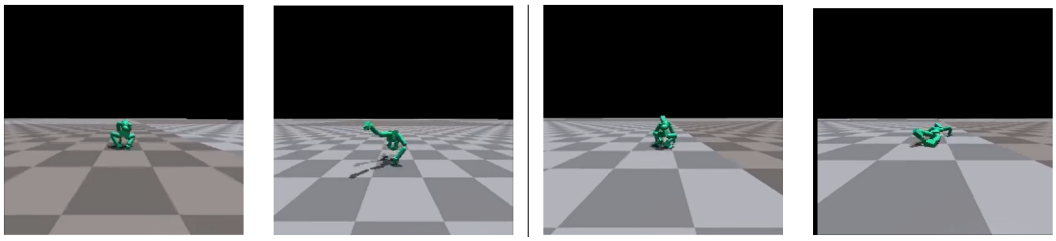
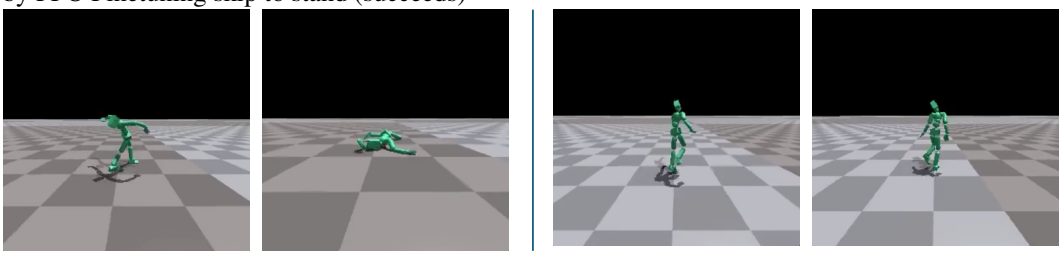


Figure 9: **Left:** 25% BC + 75% DAgger skip to stand (fails) **Right:** 25% BC + 75% DAgger followed by PPO Finetuning skip to stand (succeeds)



We hypothesize two possible explanations for this result:

- The behavioral cloning (BC) stage may lead the model to imitate the expert rather than interacting with the raw motion data, which could introduce biases.
- The BC stage might guide the model toward a local optimum—representing a different style or solution for achieving the tasks—which is suboptimal compared to the solution discovered through PPO.

Lastly, we observe that the fine-tuned policies produce qualitatively more human-like motion than their Stage 2 counterparts. That said, certain behaviors still appear unnatural or strained.

6 Discussion

6.1 Limitations

Our main limitation is the small scale of our experiment. Unlike recent works Tessler et al. (2024); Truong et al. (2024); Juravsky et al. (2024), our unified model only includes six motions. It is unclear how scaling our approach to more motions would affect performance. Additionally, we lack ablations and repeated runs for Stage 2 training. This makes the stochasticity of our training process unclear and makes it difficult to determine the optimal ratio of expert to relabeled actions. We also did not try many learning rates, which may explain why training loss plateaued early.

6.2 Broader Impact

This project provides valuable insights into imitation learning, reinforcement learning, and multi-task learning. While DAgger improved performance, it is also worth highlighting the risks of using neural network experts, which can behave unpredictably under distribution shift. A warm-up phase and including some expert data during DAgger appears essential to prevent learning from noisy labels (when experts cannot recover), but more experiments are necessary. We also observed that imitation-learned motions appear less natural—likely due to the coarse negative log-likelihood loss, compared to the richer DeepMimic reward.

Surprisingly, PPO fine-tuning from Stage 2 models was less effective than training from scratch. This raises interesting questions about the current trend in argentic LLM training – supervised learning first and then RL fine-tuning. Could the supervised pretraining not always benefit downstream RL fine-tuning?

6.3 Challenges

We faced two main challenges:

1. Working with the MaskedMimic codebase, which had many abstraction layers that made debugging motion loading difficult, resulting in incorrect BC training for much of the project.
2. Visualizing expert rollouts during DAgger, as headless training with many environments made it hard to inspect behavior. We resolved this by reducing to one environment and adding screen capture support in Isaac Gym.

7 Conclusion

In this project, we explored the imitation learning, and PPO finetuning paradigms to learn a single expert policy capable of replicating six different human motions. Based on the experiments we conducted, we find that imitation learning is somewhat successful in replicating some of the motions (success rate $\sim$ 0.5-1). On the other hand, PPO-finetuned models are able to achieve this objective successfully (success rate of 1), even when initialized from scratch. In the future, we think it would be a good idea to confirm some of the trends we observed by collecting more empirical data.

8 Team Contributions

- **Andi:** Equal contribution for training for training final models; Dataset Exploration; BC+DAgger Training Loop and Infra; and Evaluation Code
- **Eric:** Equal contribution for training final models; IssacGym Visualization; Integration with PPO Fine-tuning, parts of DAgger rollouts and expert loading
- **Silky:** Equal contribution for training final models + initial experimental runs; DAgger expert rollouts; loading of tasks for multi-task learning in DAgger.

Changes from Proposal Our high-level idea is the same as the proposal, that of training a single unified policy network for multi-action control. There are some technical changes in the experimental setting viz. the overall method, how we source the reference motion clips, evaluation metric, using reference motion instead of text for controlling the motions.

References

Advanced Computing Center for the Arts and Design. [n. d.]. ACCAD MoCap Dataset. `https://accad.osu.edu/research/motion-lab/mocap-system-and-data`

Jordan Juravsky, Yunrong Guo, Sanja Fidler, and Xue Bin Peng. 2024. SuperPADL: Scaling Language-Directed Physics-Based Control with Progressive Supervised Distillation. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH '24)*. ACM, 1–11. `https://doi.org/10.1145/3641519.3657492`

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. 2015. SMPL: A Skinned Multi-Person Linear Model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)* 34, 6 (Oct. 2015), 248:1–248:16.

Zhengyi Luo, Jinkun Cao, Alexander W. Winkler, Kris Kitani, and Weipeng Xu. 2023. Perpetual Humanoid Control for Real-time Simulated Avatars. In *International Conference on Computer Vision (ICCV)*.

Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. 2019. AMASS: Archive of Motion Capture as Surface Shapes. In *International Conference on Computer Vision*. 5442–5451.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. 2021. Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning.

Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. 2015. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342* (2015).

Chen Tessler, Yunrong Guo, Ofir Nabati, Gal Chechik, and Xue Bin Peng. 2024. MaskedMimic: Unified Physics-Based Character Control Through Masked Motion Inpainting. *ACM Transactions on Graphics (TOG)* (2024).

Takara Everest Truong, Michael Piseno, Zhaoming Xie, and Karen Liu. 2024. PDP: Physics-Based Character Animation via Diffusion Policy. In *SIGGRAPH Asia 2024 Conference Papers (SA '24)*. ACM, 1–10. `https://doi.org/10.1145/3680528.3687683`

A Visuals of Experts

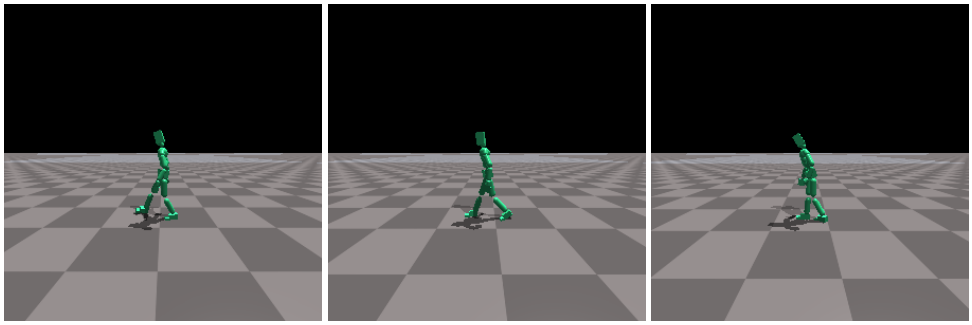


Figure 10: Single Action Expert: Walking Motion, link to wandb video

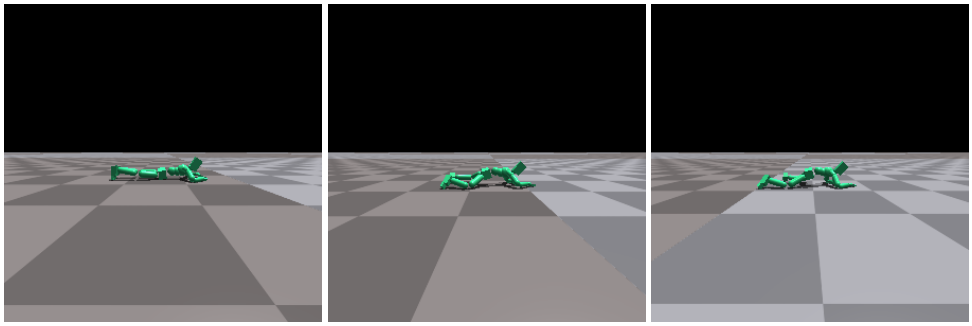


Figure 11: Single Action Expert: Crawl Motion

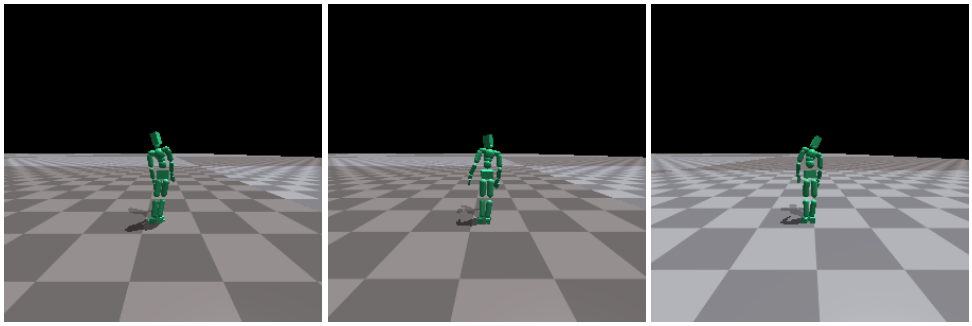


Figure 12: Single Action Expert: Swing Motion

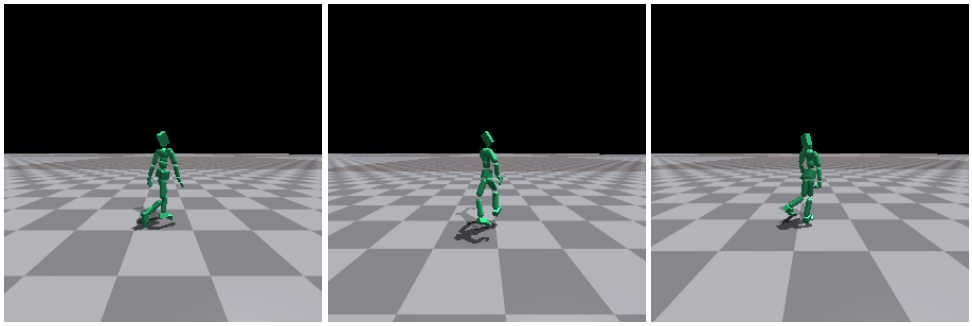


Figure 13: Single Action Expert: Skipping Portion of Stand to Skip Motion, link to wandb video

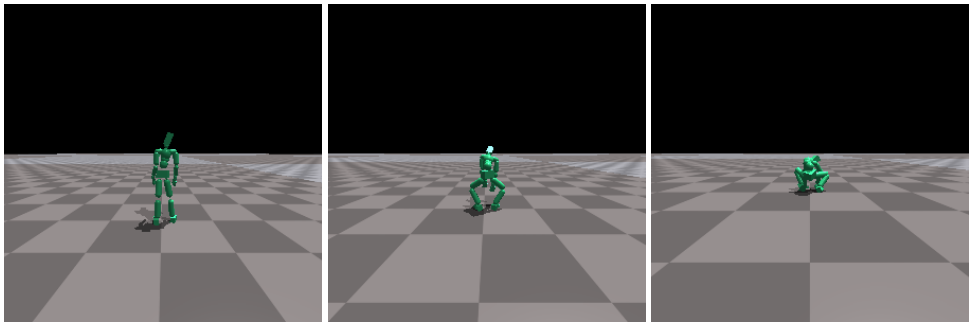


Figure 14: Single Action Expert: Crouch Pose, link to wandb video

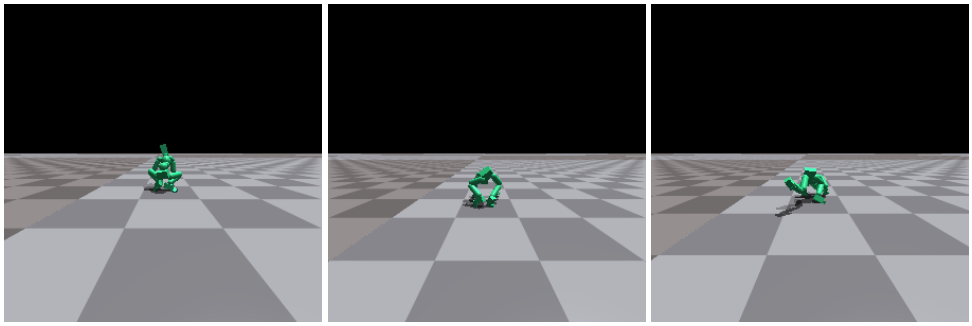


Figure 15: Single Action Expert: Crouch to Lie Pose, link to wandb video