– Independent Work Report Spring 2025 –

Learning to Walk Like Humans Do

Natalia Espinosa Dice
Adviser: Tom Griffiths

April 27, 2025

Abstract

Humanoid locomotion remains a major challenge for deep reinforcement learning (RL), requiring extensive training time while often producing unnatural, unstable gaits. Inspired by the efficiency of human motor development, this work explores two complementary strategies to better emulate that learning process: a curriculum of locomotion subtasks aligned with developmental milestones and a progressively expanding neural network architecture. We first establish baselines for individual subtasks, providing reference rewards and timestep thresholds to guide curriculum design. We then conduct a preliminary investigation into the effects of different curriculum orderings, highlighting the difficulty of transferring knowledge across tasks and suggesting directions for future improvement. In parallel, we systematically run ablation studies on depthwise and widthwise expanding networks for the standing task. Together, these experiments lay a strong foundation for future work that integrates a structured curriculum and an expanding network, with the ultimate goal of achieving more sample-efficient, human-like locomotion.

1. Introduction

1.1. Motivation

Humans learn to walk with remarkable efficiency. Within eight to eighteen months, infants progress from complete immobility to confident bipedal locomotion. In contrast, humanoid robots trained with reinforcement learning (RL) require orders of magnitude more learning experience and still struggle to produce natural gaits. This discrepancy raises an important question: what makes walking relatively tractable for humans, yet challenging for RL-trained humanoid robots?

A key distinction lies in the structured developmental timeline of human motor development. Infants do not learn to walk all at once; rather, they follow a well-defined sequence of subtasks - such as sitting, crawling and standing - that each reduce the degrees of freedom involved at a given stage.[1] From an optimization perspective, learning locomotion presents a vast search space that becomes far more tractable to navigate when segmented into smaller, progressively solvable subtasks. This developmental progression - or *curriculum* - thus helps establishes a defined roadmap toward optimal motor policies.

A second key distinction lies in the adaptive nature of the developing infant brain. Unlike the static architectures typically employed in RL systems, an infant's neural capacity grows dynamically, expanding its representational capacities over time. The principle of "starting small" suggests that early cognitive constraints - on memory, attention span and processing resources - encourage the mastery of simple foundational behaviors first, thereby providing a valuable inductive bias.[2] As neural capacity expands, these early-acquired skills form a scaffold upon which more advanced motor capabilities can be built.

Despite this insights, however, most existing RL frameworks typically treat walking as a single monolithic task and rely on static neural architectures. Although such approaches can eventually yield walking policies, they often demand enormous computational budgets or end up producing awkward, jittery gaits. Alternatively, they may depend on external scaffolds such as footstep planners or motion capture datasets, essentially shifting the burden of the problem elsewhere. A more developmentally inspired approach - one that grows neural capacity in tandem with subtask mastery along a curriculum - may better replicate the sample efficiency and fluidity observed in human motor development.

1.2. Goal

Motivated by these insights, our goal is to design a control policy for humanoid locomotion that achieves efficient, natural walking with reduced sample complexity. To this end, we propose a framework inspired by developmental psychology that leverages two complementary strategies.

First, we design a curriculum of locomotion tasks aligned with human developmental milestones, enabling the agent to master progressively more complex subtasks before attempting full bipedal walking. Second, we adopt an expanding network approach, gradually increasing the policy's network capacity in tandem with subtask mastery. By combining these two strategies, we aim to consolidate foundational skills early in training while continuously accommodating new complexity, thus establishing a more structured pathway to complex motor control.

In this semester's work, we took initial steps toward that goal by running separate experiments on each component of our overarching approach. With respect to the curriculum, we began by assessing the learnability of individual subtasks, laying a foundation for future sequencing by establishing reward and timestep thresholds. We also ran an initial exploration of subtask orderings, finding that the sequence of [prone $\rightarrow$ crawl $\rightarrow$ knees $\rightarrow$ standing with box assistance $\rightarrow$ standing] performed best. We observed limitations in transfer across subtasks, suggesting that the curriculum would strongly benefit from integration with an expanding network to reduce forgetting. With respect to the expanding network, we ran ablation studies on both depthwise and widthwise expansions for the standing subtask. Our results underscore the importance of carefully tuning the learning rate decay factor and highlight the impact of various architecture sizes. We also gained valuable insights into learning locomotion more broadly - controlling the output layer size, for instance, is important to avoid compressing high-dimensional action signals too aggressively. Overall, this body of work lays the groundwork for a more comprehensive system, one that unites a well-structured curriculum with progressive network expansion and ultimately emulates the efficient, structured learning process observed in humans.

2. Problem Background and Related Work

In this section, we first review relevant work in continual learning, which provides the theoretical foundation for our approach. We then discuss previous RL approaches to humanoid locomotion, emphasizing how our approach is unique from existing literature. Finally, we provide a brief overview of our choice of algorithm: Proximal Policy Optimization.

2.1. Continual Learning

Continual learning addresses the challenge of enabling an agent to sequentially learn multiple tasks without losing previously acquired capabilities. In practice, this means adapting the same model to handle a series of tasks $T_1, T_2, ..., T_k$ over time.[3] The core challenge in this setting is catastrophic forgetting, wherein updating model weights for a new task dramatically degrades performance on earlier tasks, as parameters critical to those skills are overwritten. A variety of methods have been proposed to mitigate catastrophic forgetting. Replay-based methods interleave new experiences with stored (or synthetic) data examples from previous tasks, effectively reminding the model of earlier knowledge. Regularization-based approaches, such as Elastic Weight Consolidation (EWC), constrain parameter updates based on how critical each parameter is to previously learned tasks. By adding a penalty term derived from a Fisher information matrix approximation, EWC anchors weights crucial for older tasks while still allowing plasticity for learning new tasks.[3]

A more architectural approach is taken by Rusu et al. [4] via progressive neural networks (PNNs), which incrementally expand network architecture by introducing a new neural column for each successive task T_k. Columns corresponding to prior tasks remain frozen, thus preserving previously learned representations, while lateral connections enable new columns to selectively incorporate earlier features. This design thus not only mitigates forgetting but also typically shows strong transfer learning, particularly in comparison to classic pretraining and finetuning approaches.[4]

Among these approaches for mitigating forgetting, PNNs are especially relevant to our work because they embody the principle of "starting small" - a developmental concept suggesting that constraining network capacity during initial learning can actually enhance learning by forcing the system to focus on mastering foundational patterns first.[2] As capacity increases over time, the learner can then build on these robust foundational representations. Elman [2] demonstrated this phenomenon in language tasks, where a recurrent neural networks succeeded in mastering recursive linguistic structures only when its memory capacity started small and gradually expanded. This parallels the way children learn to speak: beginning with simple speech before scaling up to more complex language as cognitive capacity develops.

We can also think of the starting small principle in terms of the learner's environment, through the establishment of a training curriculum. In supervised contexts, curriculum learning sequences subtasks from simple to complex to improve convergence and generalization. In continual learning, this idea extends beyond the ordering of data points within a single task to the very arrangement of tasks themselves. Bell and Lawrence [5] found that the sequence in which tasks are introduced significantly affects both average accuracy and forgetting. Specifically, "maximum distance" task sequences (based on curvature or task similarity) tended to minimize forgetting more effectively than "minimum distance" paths. Although not all selectively ordered curricula outperformed random baselines, these results highlight task ordering as a key design consideration in continual learning. Elman [2] further emphasized the utility of an incremental curriculum by training networks on linguistic inputs structured into five phases of increasing complexity. Networks trained with this curriculum demonstrated superior generalization across linguistic structures compared to those trained on the full dataset all at once. Elman noted that children are exposed to complex adult speech from the start, suggesting that the primary developmental constraint in this context is the child's neural capacity rather than environmental (or input) complexity.

In the domain of motor control, however, environmental complexity *does* follow a natural developmental progression. Indeed, developmental psychology offers a well-defined timeline of learning to walk, consisting of ten stages that each last approximately two to three months.[6] Infants begin by developing head and trunk control, first lifting their heads, pushing up with their arms and rolling over. As they gain strength, infants sit without support, crawl, stand with support and pull to stand. With greater stability, they progress to cruising, standing alone and ultimately walking alone. While this timeline is a widely recognized standard within the field, Adolph and Robinson [1] emphasize that motor learning remains highly flexible. Infants may skip stages, reach milestones at different times or adopt alternative strategies based on environmental and cultural factors. While this curriculum provides a structured template, it leaves some room for flexibility in terms of exact orderings or permutations, which we aim to explore.

Our work thus seeks to incorporate both an expanding network - in a manner akin to PNNs - and

a developmental curriculum. By adhering to the principle of starting small, we aim to define a more structured pathway through the vast search space of learning to walk.

2.2. Learning to Walk: Previous Approaches in RL

Having established the theoretical foundations of our framework, we now turn to review existing RL strategies for humanoid locomotion. Broadly, RL approaches can be grouped into three main paradigms: hierarchical frameworks, reference-based end-to-end frameworks and reference-free end-to-end frameworks.[7] We provide an overview of these paradigms, illustrating their respective trade-offs and their relationship to our own approach.

2.2.1. Hierarchical Frameworks: Hierarchical RL decomposes the control problem into at least two levels: a high-level policy responsible for long-horizon planning and a low-level policy that executes detailed motor commands.[7] This separation is particularly effective in locomotion tasks, where the high-level controller might select gait types or footstep plans while the low-level controller handles joint-level dynamics. For example, Yang et al. [8] introduce a high-level policy to modulate gait parameters (such as stepping frequency) while a low-level convex model predictive controller maintains balance. They find that gait switching (walk to trot to fly-trot) emerged naturally at different speeds, achieving significantly improved efficiency compared to manually engineered transitions. Similarly, Singh et al. [9] combines a high-level footstep planner with a learned policy that outputs joint-angle targets. Trained under a curriculum of increasing terrain complexity, the resulting controller demonstrates robust, goal-directed walking, stair-climbing and backward motion. Despite their effectiveness, hierarchical frameworks often require significant human-designed scaffolds, such as external footstep planners or carefully defined model predictive controllers. This reliance on engineered structures can limit flexibility and make it more difficult to integrate new tasks or environments.[7]

Andreas et al. [10] offers a slightly more flexible alternative by introducing symbolic “sketches” that decompose tasks into learnable subpolicies. These sketches encourage the system to discover and reuse shared structures across tasks, supporting generalization and transfer to new tasks. While

our implementation differs - namely we avoid explicit symbolic scaffolding and tackle the more complex challenge of humanoid locomotion - the underlying motivation is similar: to structure learning in a way that encourages progressive skill acquisition and reuse of prior knowledge. Thus, our approach continues to move away from the engineering overhead of hierarchical approaches, instead targeting a simpler and more unified end-to-end solution. While our use of a curriculum may bear superficial resemblance to hierarchical frameworks, our method learns a single unified policy throughout training, rather than dividing control across separate high and low-level modules.

2.2.2. Reference-Based End-to-End Frameworks: Reference-based RL approaches rely on reference trajectories, often from motion capture data, to guide the agent via imitation learning. For instance, Li et al. [11] employ imitation learning to train a joint-level PD controller for a wide range of bipedal skills - including walking, running and jumping - in both simulation and the real world. By training on a three-stage curriculum of single-task imitation, task randomization (varying speed or height targets) and dynamics randomization, they enable robust transfer from simulation to the real world. A second example by Xu et al. [12] introduces a multi-objective reinforcement learning framework that enables agents to imitate partial-body motions from multiple reference clips simultaneously. Rather than learning a full-body imitation policy, the method employs multiple discriminators, each focusing on a subset of joints. This structure allows the agent to learn modular, composable behaviors - such as walking while aiming - while jointly optimizing task-specific objectives. In a similar setup, Peng et al. [13] trains a latent skill-conditioned low-level policy on large unstructured motion datasets. A high-level controller then learns to steer the agent through this latent space, enabling composition of complex behaviors for competitive sport tasks - including running, striking and recovering from falls - without needing specifically labeled or segmented motion clips.

Overall, reference-based RL approaches reliably produce high-quality, human-like motion and excel at complex motor behaviors. However, they depend heavily on curated motion datasets and engineered pipelines, which can confine resulting policies to specific gaits, restrict its ability to explore a broader range of motions and limit adaptability in responding to unforeseen environmental

changes.[7] By contrast, our approach targets fully reference-free locomotion.

2.2.3. Reference-Free End-to-End Frameworks: Reference-free methods discover locomotion through trial and error in the absence of motion-capture trajectories, relying solely on reward signals and raw environment interactions. For instance, Ha et al. [14] address the complexities of real-world quadrupedal locomotion by automating resets and incorporating safety constraints. They train multiple tasks - forward/backward walking and turning - in a single run, using a scheduler that decides which directional task to attempt so that the robot remains in the training region. Additionally, they introduce roll and pitch limits to prevent falls. This combination drastically reduces the need for human intervention; after a few hours of on-board RL, the robot learns robust, multi-directional gaits with minimal resets.

Focusing more on simulation, Tao et al. [15] address the notoriously difficult get-up task for humanoids, where standard methods often yield unnatural or erratic behaviors. The authors propose a three-stage training pipeline: use a strong agent to discover viable get-up strategies, apply a strong-to-weak torque curriculum to refine low-energy variants of these motions and finally, perform slow-motion imitation of learned behaviors. The final policy, trained via Soft Actor-Critic, exhibits robust, more natural get-up motions and generalizes across body configurations, including agents with missing limbs or casts.

In another vein, purely mechanical objectives can also yield more natural movements. Fu et al. [16] use a reward function that punishes net torque usage and provides a small bonus for forward progress. They show in simulation and on a real-world quadruped that this approach yields canonical gaits (walk, trot, gallop) at different speeds, consistent with real animals. These emergent gaits require no pre-programmed sequences or demonstrations and extends robustly to irregular terrains.

Alongside energy, several recent works focus on enforcing symmetry in an effort to produce more natural motion. Yu et al. [17] combines a symmetry-enforcing term in the policy loss with an energy-minimizing reward. A novel “assistance“ curriculum is also introduced, wherein an external force is gradually removed as training progresses, preventing the agent from collapsing or stumbling early on. By the end of training, the agent achieves low-energy, left-right symmetric locomotion

across both biped and quadruped morphologies. A more comprehensive survey, Abdolhosseini et al. [18], compares four different approaches to enforcing symmetry: data augmentation (duplicating experiences with mirrored states/actions), an auxiliary symmetry loss in the network objective, direct phase-based reflection (splitting the gait cycle and reusing it mirrored) and a specialized mirroring architecture. They evaluate these methods across various locomotion tasks, finding that auxiliary-loss symmetry is the most consistently beneficial in achieving strong final performance and near-symmetric motion, though each approach has unique trade-offs.

Reference-free learning enables the exploration of novel solutions that might not be discovered through reference-based learning. However, it also tends to require extensive and time-consuming training while potentially struggling to master more complex tasks.[7] Furthermore, these approaches often rely on careful reward shaping (torque penalties, symmetry losses) or additional environmental scaffolds (external force assistance). In other cases, large-scale computational budgets also play a critical role in achieving state of the art results. Wang et al. [19] is a notable example, albeit in self-supervised RL, where massive model capacity dramatically boosts performance and leads to emergent behaviors, albeit at extremely high computational cost.

We position our work within the reference-free paradigm, deliberately avoiding reliance on motion capture data or the engineering overhead typical of reference-based and traditional hierarchical methods. However, our method departs from much prior reference-free research by pursuing a dual strategy of a developmentally grounded curriculum and an expanding network. We forgo heavy reward engineering - such as explicit symmetry or energy terms - and instead lean on the structure provided by developmental constraints to simplify early learning. While some prior work incorporate partial curricula (torque scaling, progressive terrain difficulty), ours is uniquely inspired by human developmental milestones and coupled with incremental expansions of network capacity. Our aim is to achieve sample-efficient humanoid locomotion with stable, human-like gaits without requiring either large-scale training infrastructure or meticulously shaped reward functions.

2.3. Algorithms

Proximal Policy Optimization (PPO) is an on-policy policy-gradient algorithm that offers a favorable balance of conceptual simplicity and strong empirical performance on continuous-control tasks. PPO alternates between collecting on-policy trajectories using the current policy and optimizing a clipped surrogate objective that constrains how much that policy can deviate from the old policy.[20]

Concretely, PPO maximizes the following clipped surrogate objective:

$$L^{\text{CLIP}}(\theta) = E_t\left[\min\left(r_t(\theta)\hat{A}_t,\ \text{clip}\left(r_t(\theta), 1-\varepsilon, 1+\varepsilon\right)\hat{A}_t\right)\right]$$

where the probability ratio $r_t(\theta)$, the estimated advantage $\hat{A}_t$ and the temporal difference residual δ_t^V are defined as:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, \qquad \hat{A}_t = \sum_{l=0}^{\infty}(\gamma\lambda)^l\delta_{t+l}^V, \qquad \delta_t^V = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$$

In complex environments, PPO can be less sample-efficient than off-policy algorithms such as Soft Actor-Critic (SAC), which exploit maximum-entropy objectives and replay buffers for better exploration.[7] Nonetheless, PPO remains popular due to its ease of use and flexibility. Our hope is that the dual strategy of a curriculum and expanding network can partially mitigate PPO's inefficiencies by simplifying early subtasks and aligning network capacity with emerging complexities. If successful, this would preserve PPO's practical advantages (on-policy updates, stable hyperparameter tuning) while narrowing the performance gap with more sophisticated off-policy methods.

3. Approach

To address the challenges outlined in previous sections and exploit the potential benefits of imposing developmental constraints, our overarching framework leverages two novel strategies: (1) a structured curriculum based on developmental milestones and (2) a progressively expanding neural

network. This semester, each idea was explored largely in isolation as we tested feasibility, with plans to build upon this work in the future.

3.1. Curriculum Inspired by Developmental Milestones

Our first strategy mirrors the motor development timeline observed in human infants, where simpler phases (prone or crawl) naturally precede more advanced behaviors (standing or walking). Rather than treating locomotion as a single monolithic challenge, we decompose it into subtasks of increasing difficulty, each focusing on simpler, incremental components of motor control. Our curriculum is strongly inspired by the developmental timeline outlined in Figure 1. In principle, splitting locomotion into subtasks reduces the degrees of freedom - and therefore the exploration complexity - required in early stages. An agent can thus first learns basic stability and partial weight-bearing before tackling the more challenging upright stance. This approach aims to foster skill reuse and minimize catastrophic forgetting when transitioning between tasks, in accordance with curriculum learning literature.

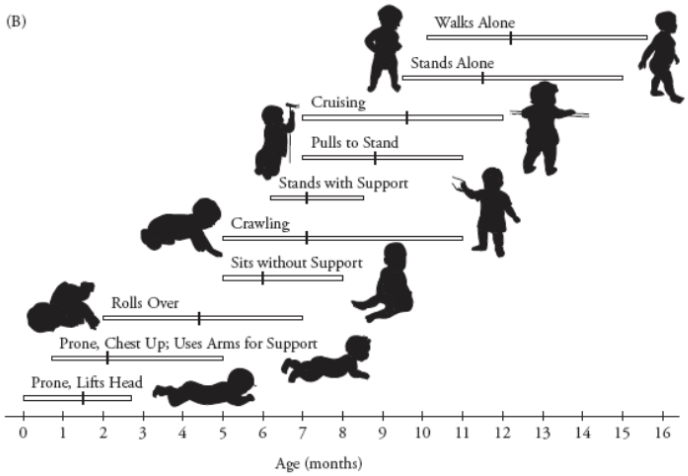

Figure 1: Developmental Motor Milestone Chart [1]

This semester, we first established performance baselines by training our model separately on each individual subtask, providing reference points for rewards and training durations. Subsequently, we began experimenting with various orderings of subtasks to investigate how task sequencing impacts learning stability. We implemented threshold-based promotions, where the agent graduates

to the next subtask once the current reward exceeds a threshold. Despite its conceptual appeal, our preliminary experiments highlight some substantial challenges. Transitioning from, for instance, a prone state to a supine or seated state fundamentally alters the agent's initial posture and feasible action sequences. From a conceptual standpoint, it thus shifts the underlying Markov Decision Process, which can complicate the process of transferring knowledge across tasks. While certain curriculum orderings improved transfer, we found that in most cases, our reward thresholds were rarely met within the maximum training time allotted to each stage. In Section 4.4, we elaborate on these challenges, offer a more detailed explanation of our results and propose potential solutions.

3.2. Expanding Network Capacity:

We also implement an expanding network strategy. Traditional RL approaches typically utilize static - and sometimes enormous - network architectures, which often struggle to learn locomotion efficiently, especially when forced to handle all complexities at once. Instead, we explore an expanding network, gradually increasing representational capacity over time.

The principle of starting small suggests that limited early capacity may help humans master simpler behaviors first, without being overwhelmed by complex tasks. As complexity increases, the brain "expands" its effective capacity, building upon already-stabilized representations. Translating this idea to RL, we initially train a smaller policy network, then add layers or widen existing layers to handle more demanding phases of the task. After expanding, we slow down the learning rate of previous weights to prevent forgetting. While Rusu et al. [4] discuss "freezing" previous layers altogether, we explore various learning rate decay factors in the range [0.1, 0.3, 0.5, 0.7, 0.9].

This semester, we ran systematic ablations on the standing task varying initial network size, the expansion direction (width or depth) and the learning rate decay. While our tested expanding networks lag behind static baselines in terms of final reward, these ablations offer insights into potential levers to pull on in the future for further improvement, as detailed in Section 4.5.

3.3. Toward A Unified Framework

Although each component was tested separately this semester, our ultimate objective is to merge them into a single, cohesive training pipeline. In principle, as the agent progresses through the curriculum, the network itself will scale up, leveraging earlier subtask knowledge without overwriting it. Incorporating an expanding network may therefore help mitigate the challenge of "switching" tasks and balance maintaining plasticity for learning new tasks while mitigating forgetting. While key challenges still remain to be overcome in each setting, this semester's work offers valuable insights that will inform future work, in which we aim to continue refining the curriculum and expansion protocols into a unified training pipeline.

4. Implementation

Below, we summarize our system design, environment details and experimental procedures for both the curriculum and expanding-network experiments.

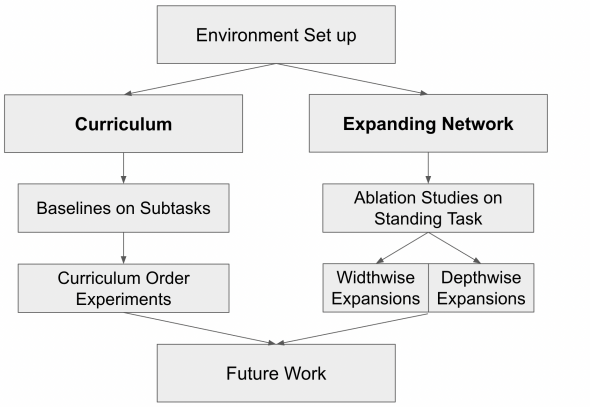

Figure 2: Implementation Overview

4.1. Environment Setup

We build on Gym's MuJoCo-based `HumanoidGetUp` environment with the following modifications.[21]

- **Initial Configurations:** Each subtask (prone, supine, seated, crawl, knees, standing with boxes, standing) uses a different joint configuration loaded from a keyframes file.

- **Observation/Action Spaces:** Observations include joint angles, velocities, inertias and contact forces. Actions are continuous joint torques.
- **XML File:** We use Gymnasium's humanoid.xml file, which has 23 degrees of freedom. For the standing with boxes task, we modify the file to include two boxes to be used for support.
- **Reward Function:** We adopt a minimal reward structure incentivizing an upright torso (via head height) and penalizing control effort:

$$\text{reward} = w_{\text{upright}} \times \text{head\_pos} - w_{\text{ctrl}} \times \sum(\text{control})^2$$

- **Environment Preprocessing:** We use vectorized environments. Observations and rewards are normalized using the VecNormalize library from StableBaselines and wrapped with VecMonitor for logging.
- **Episode Termination**: Triggered if an agent falls outside a valid z-range or reaches the maximum episode length of 2000 steps. Valid z-ranges depend on on the goal height of the humanoid for the task at hand. For laying down positions, there is no early termination for invalid z-ranges.
- **Noise and Variation:** We add small random perturbations to initial joint poses. For crawling or kneeling, additional randomization in arm and knee angles encourages robustness.

4.2. Algorithms and Libraries

We use Proximal Policy Optimization (PPO) from `StableBaselines3 (v2.5.0)`.[22] Our choice of PPO is deliberate: its simplicity and stability make it a strong candidate for integration with our developmental strategies. We hypothesize that, by introducing curriculum and capacity growth, we may compensate for some of PPO's sample inefficiencies. Training uses GPU for the policy and CPU for MuJoCo simulation, with 8 parallel environments. We track progress via Weights & Biases for logging, while hyperparameter tuning is conducted using Optuna.[23] [24]

4.3. Baselines for Individual Subtasks

In order to establish the feasibility of learning each developmental stage in isolation, we trained a series of single-task baselines using PPO. These tasks correspond to the potential stages of our proposed curriculum: prone, supine, seated, crawl, knees, standing with boxes and standing. Training and evaluating a policy on each subtask separately allows us to (1) confirm solvability with sufficient timesteps and (2) establish performance thresholds for later curriculum experiments.

4.3.1. Hyperparameter Search with Optuna We conducted an initial hyperparameter search for each subtask using the Optuna library.[24] Each search was comprised of 50 independent trials, where each trial was trained for up to 1 million timesteps (and sometimes fewer for simpler tasks such as prone, supine and crawl).

Table 1: Hyperparameter Search Space for Baseline Subtasks

Parameter	Search Space
Learning Rate (α)	Log-uniform in $[10^{-5}, 10^{-3}]$
Number of Steps (n_{steps})	{1024, 1536, 2048, 2560, 3072, 3584, 4096}
Batch Size	{32, 64, 128, 256}
Use SDE	{True, False}
Discount Factor (γ)	{0.95, 0.99}
Number of Epochs (n_{epochs})	{1, 5, 10}
Entropy Coefficient	{0.0, 0.005}
Use Weight Decay	{True, False}; if True, log-uniform $[10^{-5}, 10^{-2}]$
Architecture Scale	{1, 2, 3, 4}

The policy network architecture depended on the integer parameter `architecture_scale`, which scaled the first two hidden layers by multiples of 16, while the final layer was held fixed at 16 ([16, 16, 16], [32, 32, 16], etc.). All experiments were run with $n_{\text{envs}} = 8$ parallel environments to accelerate data collection. The remaining standard PPO parameters remained fixed at reasonable, well-established values, including `clip_range = 0.2`, `vf= 0.5`, `gae_lambda = 0.95` `max_grad_norm = 0.5` and `normalize_advantage` = True . The best trial was chosen according to mean final episode reward during final evaluation across 5 episodes after training.

4.3.2. Extended Training Runs Once we identified the best hyperparameters for each subtask, we conducted longer training runs with the aim of reaching near-convergence. Table 2 summarizes the timesteps used for each subtask alongside the intended final goal position.

Table 2: Extended Training Timesteps for Each Subtask Baseline

Subtask	Timesteps	Goal / Final Position
Prone, Supine	2M	Lift head/torso from floor
Seated	3M	Maintain balance in a seated pose
Crawl	3M	Push back to a kneeling position
Knees	3M, 8M	Balance on knees, potentially initiate standing
Standing with Boxes	5M	Balance upright with box assistance
Standing	10M	Maintain upright stance without support

4.3.3. Monolithic Prone $\rightarrow$ Standing Baseline As a final comparison, we experimented with a single-task environment that starts the agent in a prone position and asks it to reach a stable standing pose. Despite extending training to 10 million timesteps, the agent could not reliably move out of the prone position. Late in the semester, we introduced Random Network Distillation (RND) as an intrinsic reward signal to encourage exploration.[25] This vastly improved learning and will be a useful addition to PPO to incorporate in future work.

4.4. Initial Curriculum Experiments

Inspired by the per-subtask successes of Section 4.3, we next explored a threshold-based curriculum. In this setup, the agent begins training in one environment and progresses to the next subtask only after surpassing a designated reward threshold based on the subtask baselines from the previous section. This process continues sequentially through increasingly difficult subtasks until it reaches the final standing task. The motivation is to gradually expand the agent's state-space complexity while building on prior capabilities.

We leveraged the best-performing hyperparameters for the standing baseline to remain consistent with the final goal of upright balance, though we also experimented with using a larger hidden size of 128 in addition to 32. For each curriculum step, the agent collects on-policy rollouts in the current

subtask environment. We periodically evaluate the agent, and if its average episodic reward exceeds a threshold (predetermined from single-task benchmarks), we switch the training environment to the next subtask and continue. We also placed a maximum limit on training timesteps (8-10 million) on any given intermediate stage, in order to ensure that the full training run could proceed. For the final standing task, we expanded that timestep limit to 10-15 million. While the choice of threshold was derived from the final performance of single-subtask runs, we also experimented with slightly lower or higher thresholds to see if this would yield faster progression or greater stability.

Throughout the semester, we tested numerous permutations of the subtask order, primarily to investigate whether certain transitions might be more natural than others. Table 9 details these orderings, along with the architecture hidden size used. For each ordering, we also experimented with different reward goals and levels of noise. While we originally intended to conduct more rigorous testing (across architectures, random seeds and configurations), time constraints limited our experimentation to merely an initial search. In the future, we intend to more robustly explore these levers, in addition to the aforementioned addition of RND. We also intend to apply our expanding network strategy, once finalized.

Table 3: Curricula Orders and Architectures Tested

Curriculum Order	**Architecture Hidden Size**
Crawl → Knees → Standing with Boxes → Standing	32 and 128
Crawl → Knees → Standing	128
Knees → Standing	32 and 128
Prone → Crawl → Knees → Standing with Boxes → Standing	32 and 128
Prone → Supine → Seated → Knees → Standing	32 and 128
Standing → Standing with Boxes → Knees → Crawl (reverse)	32
Supine → Knees → Standing	32 and 128
Supine → Seated → Knees → Standing with Boxes → Standing	32 and 128
Prone → Crawl → Knees → Standing	32 and 128
Supine → Seated → Knees → Standing	128

4.5. Network Expansion Experiments on Standing Task

In addition to our static-architecture baselines, we conducted a large suite of experiments to investigate whether progressively expanding the policy network could improve learning efficiency for the standing task. We focused on standing alone because we lacked a reliable "prone to standing" baseline, as dicussed above.

4.5.1. Expansion Protocol and Hyperparameters Each experiment begins with a smaller network, which is trained for a fixed number of timesteps. Subsequently, the network expands either by increasing the depth (adding a new layer) or the width (increasing the size of existing layers). This process repeats until the target architecture is reached. For example, in a depth expansion from [32, 32] to [32, 32, 32] and finally [32, 32, 32, 32], one layer is added at each step. In a width expansion, units in each layer double in size at each step (ex: $[4, 4, x]$ to $[8, 8, x]$, $[16, 16, x]$ and so forth). Note that due to the implementation of our expanding network, the last layer is always held constant. Each phase of the expanded network trains for double the number of timesteps as the previous. For depthwise expansions, this consisted of [1M, 2M, 4M] timesteps for a total of 7 million. For widthwise expansions, this consisted of [0.25M, 0.5M, 1M, 2M, 4M] timesteps for a total of 8 million.

After expansion, we preserve old weights by slowing down the learning rate on previous layers. In particular, we varied the decay factor among $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. The goal is to see if reducing the learning rate upon expansion mitigates destructive interference - or forgetting - when new parameters are introduced. We reuse the best PPO hyperparameters identified in our single-task standing baseline, ensuring that only the network architecture and the learning rate decay factor change. We compare these final expanded models against static baselines of identical final architecture to assess whether incremental growth offers advantages in sample efficiency or final performance.

4.5.2. Static Architecture Baselines Before testing expansion, we established comprehensive static baselines with 2, 3 or 4 hidden layers of various widths and output layer size. We train each architecture on 5 seeds, thus yielding 150 trials total. These baselines provide a reference when

evaluating the performance of expanded networks that end up with, for instance, $(64, 64, 64)$ or $(128, 128, 32)$ as their final shape.

Table 4: Static Architecture Baselines

2 layers	3 layers	4 layers
$(4, 4)$	$(4, 4, 4)$	$(4, 4, 4, 4)$
$(8, 8)$	$(8, 8, 8)$	$(8, 8, 8, 8)$
$(16, 16)$	$(16, 16, 16)$	$(16, 16, 16, 16)$
	$(32, 32, 16)$	$(32, 32, 32, 16)$
$(32, 32)$	$(32, 32, 32)$	$(32, 32, 32, 32)$
	$(64, 64, 16)$	$(64, 64, 64, 16)$
	$(64, 64, 32)$	$(64, 64, 64, 32)$
$(64, 64)$	$(64, 64, 64)$	$(64, 64, 64, 64)$
	$(128, 128, 64)$	$(128, 128, 128, 64)$
$(128, 128)$	$(128, 128, 128)$	$(128, 128, 128, 128)$

4.5.3. Depthwise Expansion Experiments For depthwise expansion, we tested two principal starting shapes, adding one layer at each step, as detailed in Table 5. Each path required $5 \times 5 = 25$ trials (5 seeds $\times$ 5 learning rate decays), for a total of 50 trials. We compared the final expanded networks with their static counterparts trained from scratch.

Table 5: Depth Expansion Paths

Expansions
$[32, 32] \rightarrow [32, 32, 32] \rightarrow [32, 32, 32, 32]$
$[64, 64] \rightarrow [64, 64, 64] \rightarrow [64, 64, 64, 64]$

4.5.4. Width Expansion Experiments For width expansion, we systematically increased layer sizes, doubling at each stage, until reaching a target width. We keep the output layer fixed but test various sizes. In Table 6, we list the paths tested, each evaluated under 5 seeds and 5 learning rate

decays for a total of 25 runs per path, or 300 trials total. We then compare each final network to a static baseline of the same dimensions.

Table 6: Width-based Expansion Paths

Expansions
$[4, 4, 16] \rightarrow [8, 8, 16] \rightarrow [16, 16, 16] \rightarrow [32, 32, 16] \rightarrow [64, 64, 16]$
$[4, 4, 32] \rightarrow [8, 8, 32] \rightarrow [16, 16, 32] \rightarrow [32, 32, 32] \rightarrow [64, 64, 32]$
$[4, 4, 64] \rightarrow [8, 8, 64] \rightarrow [16, 16, 64] \rightarrow [32, 32, 64] \rightarrow [64, 64, 64]$
$[4, 4, 4, 16] \rightarrow [8, 8, 8, 16] \rightarrow [16, 16, 16, 16] \rightarrow [32, 32, 32, 16] \rightarrow [64, 64, 64, 16]$
$[4, 4, 4, 32] \rightarrow [8, 8, 8, 32] \rightarrow [16, 16, 16, 32] \rightarrow [32, 32, 32, 32] \rightarrow [64, 64, 64, 32]$
$[4, 4, 4, 64] \rightarrow [8, 8, 8, 64] \rightarrow [16, 16, 16, 64] \rightarrow [32, 32, 32, 64] \rightarrow [64, 64, 64, 64]$
$[8, 8, 64] \rightarrow [16, 16, 64] \rightarrow [32, 32, 64] \rightarrow [64, 64, 64] \rightarrow [128, 128, 64]$
$[8, 8, 128] \rightarrow [16, 16, 128] \rightarrow [32, 32, 128] \rightarrow [64, 64, 128] \rightarrow [128, 128, 128]$
$[8, 8, 8, 64] \rightarrow [16, 16, 16, 64] \rightarrow [32, 32, 32, 64] \rightarrow [64, 64, 64, 64] \rightarrow [128, 128, 128, 64]$
$[8, 8, 8, 128] \rightarrow [16, 16, 16, 128] \rightarrow [32, 32, 32, 128] \rightarrow [64, 64, 64, 128] \rightarrow [128, 128, 128, 128]$

4.6. Evaluation Metrics

To comprehensively assess our two strategies, we utilize the following metrics during evaluation. For our curriculum experiments, we focus mainly on final evaluation reward, sample efficiency and qualitative video analysis. For the expanding network experiments on the standing task, we additionally consider center of mass stability, energy expenditure and wall clock training time.

- **Final Evaluation Reward:** We record the episodic return after training converges, averaging over five evaluation episodes. Higher rewards correlate with maintaining an upright posture and minimizing control costs.
- **Sample Efficiency:** We measure how many timesteps each method requires to reach the above reward. This indicates whether expansion strategies can accelerate early learning relative to static counterparts.

- **Wall-Clock Time:** Larger or deeper networks can require more computation per update. We compare total training time across models, noting that our expanding network strategy might require the same number of timesteps but run faster than a static architecture trained at full size.
- **Energy Expenditure:** We calculate per-timestep energy usage with the following formula:

$$\text{Avg. Energy per Timestep} = \frac{\sum |\text{joint\_torques} \times \text{joint\_velocities}| \, \Delta t}{\text{Total Timesteps}}.$$

- **COM Stability:** We approximate stability by measuring the agent's center of mass (COM) relative to its feet. A larger COM distance indicates less stable posture and an increased likelihood of falls.

$$\text{Avg. COM Stability} = \frac{1}{T} \sum_{t=1}^{T} \|\text{COM}_{xy}(t) - \text{FootCenter}_{xy}(t)\|.$$

- **Training Curves:** We log training curves (reward vs. timesteps) for each seed in Weights and Biases. These curves are useful for understanding how and to what extent the model retains previously learned knowledge after each expansion - or forgetting.
- **Qualitative Video Analysis:** Finally, we record videos to qualitatively assess final performance.

5. Evaluation

In this section, we detail the results of the baseline subtasks, our initial exploration of the curriculum and our expanding network ablation studies on depth and width.

5.1. Baselines for Individual Subtasks

Table 7 summarizes the best hyperparameters found via a 50-trial Optuna search for each of the seven subtasks, along with the final evaluation rewards and qualitative "video" observations. While we also searched over the `use_SDE` parameter, all subtask searches resulted in False. Even after relatively short trials (not exceeding 1 million), all seven subtasks demonstrate substantial progress towards their subtask goals.

Table 7: Results of 50 Trial Optuna Search on Subtask Baselines

Pose	Steps	LR	N steps	Batch size	γ	N epochs	Ent coef	Weight decay	Arch scale	Reward	Video
Prone	0.3M	1.00×10^{-4}	3072	128	0.95	5	0.005	7.88×10^{-4}	1	772.71	Lifts head and torso
Supine	0.2M	2.29×10^{-5}	2048	64	0.99	10	0.005	1.68×10^{-3}	4	1044.59	Sits up using arms
Seated	0.75M	6.37×10^{-4}	3072	128	0.95	10	0	8.93×10^{-4}	1	1404.50	Holds seated pose
Crawl	0.3M	6.36×10^{-4}	2560	32	0.99	1	0	0	1	955.20	Sits back on knees
Knees	1M	4.59×10^{-5}	1536	32	0.95	10	0.005	1.13×10^{-5}	2	1760.00	Balances on knees
Standing boxes	1M	2.58×10^{-4}	2560	128	0.99	10	0.005	6.19×10^{-4}	4	2377.00	Stands with boxes
Standing	1M	1.59×10^{-4}	3072	64	0.99	10	0.005	3.33×10^{-4}	2	1046.49	Falls to knees

Reward is calculated by averaging over 5 evaluation episodes after completing training. Arch scale is a constant that we multiply by 16 to get the hidden size of our architecture, ie $[16*x, 16*x, 16]$.

5.2. Extended Training Runs on Subtasks

Although the prior subtasks trials in Table 7 demonstrated promising results with relatively short training durations, we also conducted longer training runs using the best hyperparameter configurations obtained from this search. Table 8 details the final evaluation rewards and qualitative observations for each subtask, trained anywhere from 2 million to 10 million timesteps depending on the complexity of the task. Most tasks benefited from longer training time but with varying degrees of success. For example, the crawl subtask improved from a reward of 955.20 at 0.3M timesteps to 1315.90 at 3M timesteps. Similarly, the knees subtask also increased between the initial 1M and final 8M run, though interestingly, the intermediate 3M run began trying to stand more, fell more frequently and ultimately had a lower intermediate reward. However, the prone,

supine and standing with boxes subtasks decreased in reward, though all still managed to perform the desired subtask goal at least for a subset of the evaluation episodes' length. We show the prone rewards across training as an example in Figure 3. Figure 4 shows the standing rewards across training, showing a not quite stable but at least overall increasing trend over time. Note that because we did not average across multiple training seeds, the graphs are more unstable.

Table 8: Extended Training Run Results on Subtasks

Start Position	Timesteps	Final eval reward	Video description
Prone	2M	225.96	Raises torso, kicks legs, ends with head slightly lifted
Supine	2M	365.61	Violently sits up, overshoots and leans forward
Crawl	3M	1315.90	Pushes back to kneel; attempts upright kneel
Seated	10M	1436.58	Remains seated, leans on leg to lift head higher
Knees	3M	232.34	Balances on knees, tries to stand, then falls
Knees	8M	1967.44	Maintains upright kneel; no stand attempt
Standing boxes	5M	1939.62	Uses boxes for balance, slides down and occasionally falls
Standing (hidden 32)	10M	1750.86	Falls to knees and maintains balance

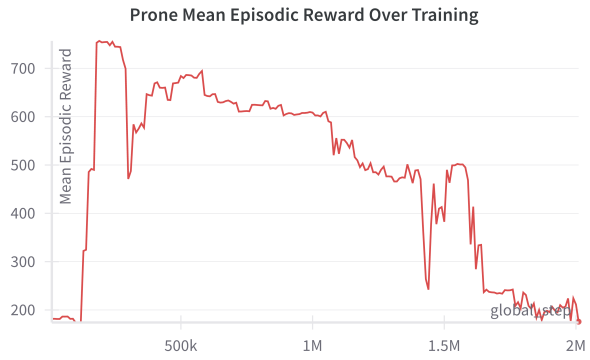

Figure 3: Mean Evaluation Reward Across Training for Prone Subtask (2M)

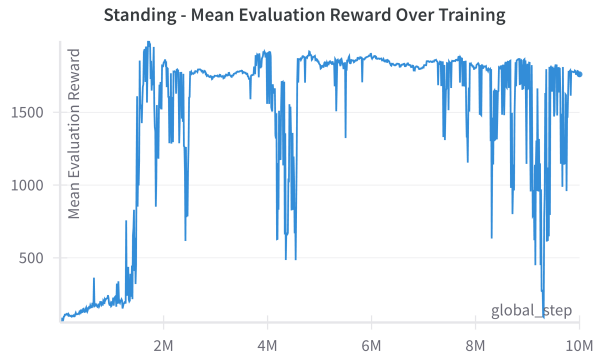

Figure 4: Mean Evaluation Reward Across Training for Standing Subtask (10M)

5.3. Monolithic Prone → Standing Baseline with RND

Establishing a monolithic prone → standing baseline is a useful reference point for our future curriculum experiments, in order to see if utilizing a curriculum improves sample efficiency and motion quality. However, as shown in Figure 3, our prone baseline was unable to move out of the prone position and even deteriorated in reward over longer training. Thus, late in the semester, we also experimented with applying RND, in an effort to encourage our agent to explore more. As shown in Figure 5, adding RND enabled much more exploration and resulted in much better training curves. Based on qualitative analysis of videos, the agent was able to reach a near-sitting upright position by 10M timesteps and a fully upright sitting position by 15M timesteps. One noticeable downside to this approach, however, is that it dramatically increased the wall clock time of training, taking nearly 22 hours to train. However, we suspect that we may be able to update the RND network less frequently (or in larger batches) and thus bring down the computational cost. While the agent still fell short of fully standing, these results point to RND as a useful tool to help encourage exploration, one we will certainly utilize in the future.

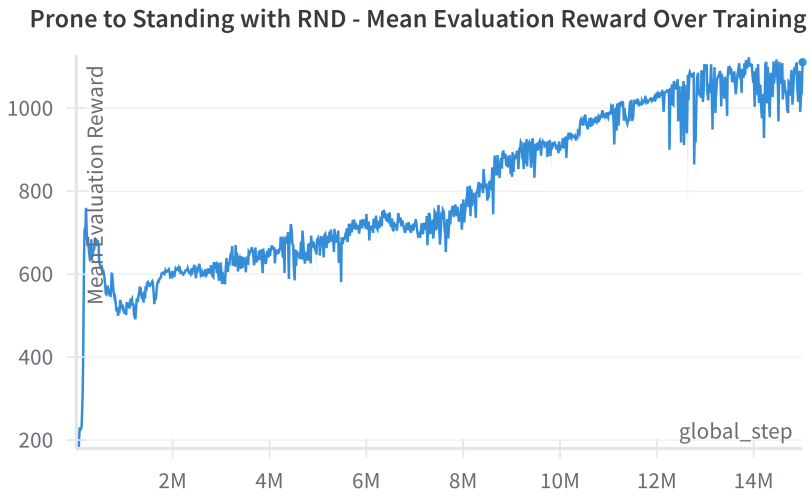

Figure 5: Monolithic Prone to Standing Baseline with RND

5.4. Initial Curriculum Experiments

Table 9: Initial Curricula Experiments Results By Reward and Length of Stages

Curriculum	Hidden	S1 Rew/Steps	S2 Rew/Steps	S3 Rew/Steps	S4 Rew/Steps	Final Standing Rew/Steps
K → S	32	38 / 8M	–	–	–	78 / 12M
K → S	128	23 / 8M	–	–	–	81 / 12M
SU → K → S	32	−208 / 8M	17 / 8M	–	–	68 / 13M
SU → K → S	128	800 / 4.5M	11 / 8M	–	–	106 / 13M
C → K → S	128	10 / 8M	31 / 8M	–	–	100 / 13M
P → C → K → S*	**32**	**459 / 1M**	**530 / 2M**	**44 / 2M**	–	**938 / 10M**
SU → SE → K → S	128	−18 / 8M	23 / 8M	31 / 8M	–	96 / 13M
C → K → SB → S	32	408 / 1M	39 / 8M	1520 / 6.5M	–	90 / 13M
C → K → SB → S	128	14 / 8M	221 / 5M	90 / 8M	–	94 / 13M
S → SB → K → C**	32	81/10M	74 / 10M	40 / 10M	–	16 / 15M
P → C → K → SB → S	**32**	**358 / 0.5M**	**471 / 3M**	**43 / 8M**	**101 / 8M**	**1570 / 13M**
P → C → K → SB → S	**128**	**454/0.5M**	**433 / 1M**	**33 / 8M**	**33 / 8M**	**1618 / 13M**
P → SU → SE → K → S	32	404 / 0.5M	−8.8 / 8M	526 / 8M	55 / 8M	281 / 13M
P → SU → SE → K → S	128	457 / 0.5 M	−132 / 8M	310 / 8M	27 / 8M	172 / 13M
SU → SE → K → SB → S	32	−211 / 8M	494 / 8M	24 / 8M	63 / 8M	63 / 13M
SU → SE → K → SB → S	128	580 / 3M	331 / 8M	11 / 8M	101 / 8M	114 / 13M

* For this run, we restricted intermediate stages to a maximum of 2M timesteps.

** For this run, we tried a reverse curriculum, beginning with the standing task and ending in prone.

We performed a small set of curriculum experiments, summarized in Table 9. Each row corresponds to a particular ordering of subtasks and a chosen network width ("Hidden"). We use abbreviated labels for each subtask: P for prone, SU for supine, SE for seated, C for crawl, K for knees, SB for standing with boxes and S for standing. The table columns indicate the average final reward on the current task and total training steps for each stage in the curriculum ("S1 Rew/Steps," "S2 Rew/Steps," etc.), as well as the final stage standing performance ("Final Standing Rew/Steps").

Because some curriculum orderings are comprised of fewer stages than others, some entries in rows are left blank, designated by a dash. We set a maximum number of timesteps for each stage (8M for intermediate stages, 12-15M for the final standing task). If the agent failed to surpass a predetermined reward threshold (based on subtask results) within this time limit, we moved on to the next subtask. This was largely to ensure that training could finish within 24 hours, but we found that varying this upper limit had an impact on performance. Because these runs were computationally expensive, we did not exhaustively vary hyperparameters architectures, maximum timesteps or reward thresholds, and we also did not run across sufficient seeds. Even so, our preliminary results reveal several noteworthy trends:

5.4.1. Task Ordering and Transfer

Our experiments tentatively suggest that less time spent on intermediate tasks correlates with better increased performance on the final standing stage. Several trials show that once a moderate reward threshold is reached - or once the stage times out on shorter maximum time limits - moving on to the next subtask sooner tends to yield higher rewards. We visualize this trend in Figure 6, which plots the final standing reward against the average length of intermediate stages. One explanation is that prolonged training on a simpler posture may reduce the policy's plasticity for future tasks, effectively overfitting the agent to an early posture. In contrast, switching tasks earlier preserves more flexibility for adapting to increasingly difficult poses.

However, the duration of these intermediate stages also reflects the effectiveness of the curriculum ordering itself. A well-structured sequence of tasks should enhance transfer, thus requiring fewer timesteps per stage. The importance of curriculum ordering is exemplified by our *reverse* curriculum trial, which progresses in the opposite direction of the timeline defined by developmental psychology with a curriculum of S $\rightarrow$ SB $\rightarrow$ K $\rightarrow$ C. This trial produced the lowest final standing reward of 16 while requiring 45M timesteps in total, indicating that an "unintuitive" task sequence hinders efficient transfer.

Together, these results highlight a key tradeoff between retaining enough plasticity to learn new tasks and avoiding forgetting of old ones. When an agent fails to effectively transfer knowledge from

earlier stages, the curriculum can become more of an obstacle than a catalyst for learning. From these observations, we identify three key directions for future work. First, curriculum ordering has a strong impact on both final performance and sample efficiency and should be more rigorously tested. Second, reward thresholds should continue to be refined. While we based our reward thresholds on the baseline subtask results, they may be too high in a curriculum setting, causing the length of intermediate stages to increase and negatively impact final performance. Finally, the overall need to improve transfer suggests incorporating our expanding network, which would enable the model to increase capacity as it encounters new task while still preserving knowledge acquired in previous stages.

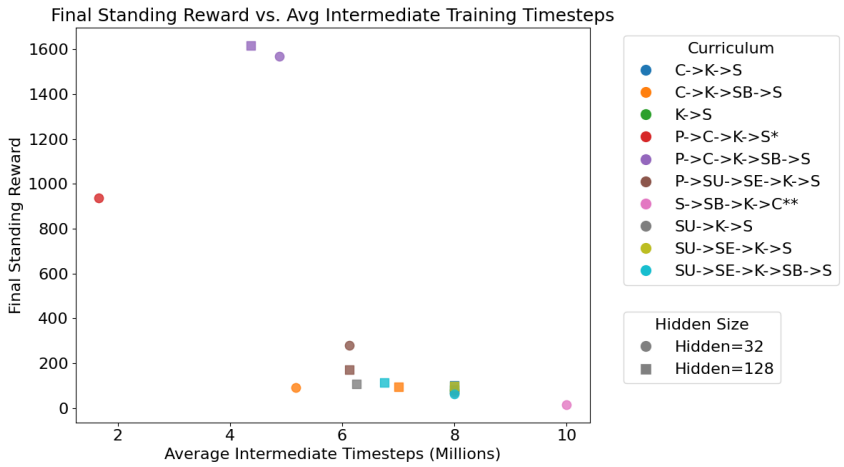

Figure 6: Final Standing Performance By Average Length of Intermediate Stages

5.4.2. Network Architecture

As illustrated in Figure 7, widths of 128 slightly outperformed widths of 32, implying that a slightly bigger model might be able to exploit increased representational power to better handle a multistage curriculum. The performance difference is not particularly large, but continuing to vary architecture sizes may be useful in the future. Furthermore, throughout all of our curriculum experiments - from baseline subtasks to these curriculum ordering trials - we utilized a final output layer size of 16. We suspect that this bottleneck may limit the policy's expressiveness. From an information-theory standpoint, a smaller output layer constrains the amount of information the network can pass from

the final hidden layer to the policy distribution parameters, thus reducing the ability to represent nuanced control strategies. Put simply, by restricting the output space of our policy, we forced a compression of the action space to just 16 outputs. In the future, we plan to experiment with increasing the output dimensionality to test whether this improves our curriculum results.

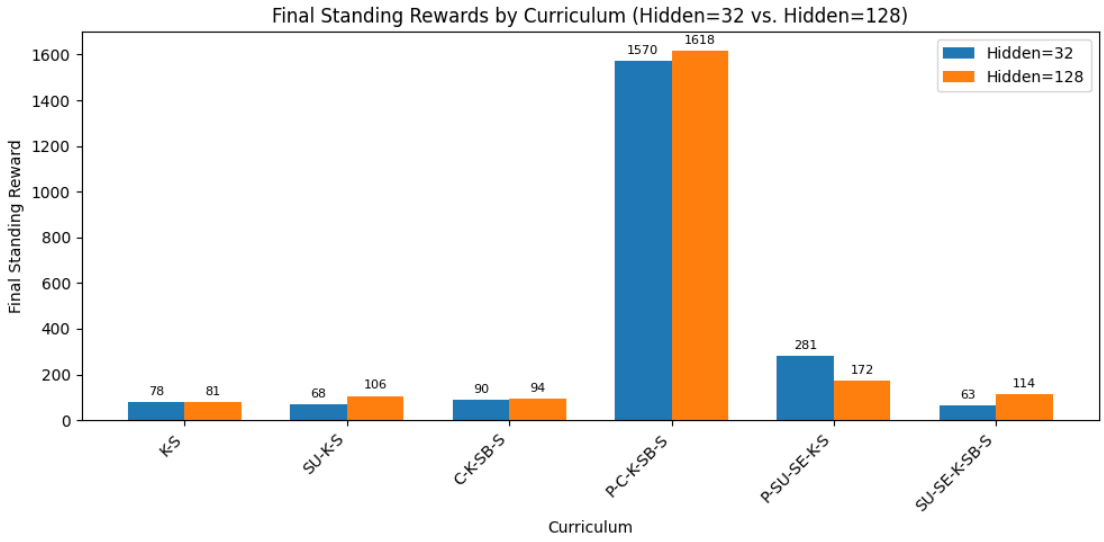

Figure 7: Final Standing Rewards by Width of Architecture

5.4.3. Negative Rewards Finally, we observe a few instances in which negative rewards reflect inactivity or poor early policies, particularly in the supine position. In these scenarios, the agent's lack of effort (and accumulating control costs) leads to net negative returns. We find this particularly surprising in trials where the supine position was the first stage, as there is no potential effect of poor transfer in these cases. However, we noted in the above section (Table 1 vs Table 2) that the supine subtask reward regressed over longer training times. In hindsight, it appears that using too high of an initial reward threshold, especially considering we used hyperparameters tuned for the standing task, likely caused prolonged training in the supine stage, causing rewards to regress dramatically. This suggests future work in continuing to refine reward thresholds and incorporating RND, which greatly improved learning stability, as discussed in Section 5.3.

5.5. Expanding Network - Depthwise Ablation Studies

5.5.1. (32, 32) $\rightarrow$ (32, 32, 32) $\rightarrow$ (32, 32, 32, 32) Expansion Path

Table 10 highlights the key quantitative findings of our depthwise expansion. Notably, the baseline - which trains a (32, 32, 32, 32) network from scratch for 10 million timesteps - achieves the highest final reward at 1953.12. In contrast, all of the expanding networks reach significantly lower final rewards ranging from 288.94 to 814.85. A closer look at the average distance between the center of mass and the center of the humanoid's feet in Figure 8 further underscores the performance gap. The baseline's COM displacement remains consistently small (around 0.1–0.2m), indicating better upright posture and control. By contrast, all expanding runs exhibit rapid, early spikes in COM displacement (to 0.4–0.5m or more), which suggests frequent falls early on in evaluation episodes.

Table 10: Comparison of $(32, 32) \rightarrow (32, 32, 32) \rightarrow (32, 32, 32, 32)$ sized expanding network to a static $(32, 32, 32, 32)$ sized baseline. All numbers are means over the five training seeds, upon each of which 5 evaluation episodes were run.

Config	LR-decay	Training Wall Time (hr)	Training Timesteps	COM*	Energy**	Final Evaluation Reward
Baseline	-	5.98	10M	0.148	18.16	**1953.12**
Expanding	0.1	2.98	[1M, 2M, 4M]	0.461	8.42	288.94
Expanding	0.3	2.97	[1M, 2M, 4M]	0.446	4.04	305.13
Expanding	0.5	3.07	[1M, 2M, 4M]	0.423	6.29	**814.85**
Expanding	0.7	3.02	[1M, 2M, 4M]	0.531	6.99	497.83
Expanding	0.9	2.97	[1M, 2M, 4M]	0.453	5.69	401.88

* Calculated as the average xy distance between the COM and the center of the humanoid's feet per evaluation timestep.

** Calculated as the average energy expenditure per evaluation timestep.

Due to the poor performance of the expanding networks, we cannot reliably compare energy expenditures, as the former often remained motionless after falling. We also cannot reliaby compare wall clock time, as the expanding networks exhibit shorter episode durations during training due to more frequent falls and poorer overall performance. However, given that these networks are not at

full size for the entire duration of training, we suspect that even with better performance, expanding networks would still consume less wall clock time than full sized static networks.

Among the expanding runs, the best expanding network used a learning rate decay of 0.5 to obtain the highest final reward of 814.85 and the lowest COM distance of 0.423. Figures 9–11 depict the training curves across the three expansion stages for this best-performing expansion, all averaged over 5 seeds. During the first stage, the mean episodic reward rises to roughly 120 by the end, suggesting that the initial (32, 32) network learns a rudimentary policy within the first million steps. After introducing a third layer (making the network (32, 32, 32)), we see an increase in reward up to around 160. This improvement reflects the newly added capacity, though the policy remains well below baseline performance at the same number of steps. In Figure 11, the network expands to its final size of (32, 32, 32, 32). By around 3.5-4M timesteps at this stage (or 7.5-8M total), rewards climb further, surpassing 300 and briefly spiking toward 500. However, even at the conclusion of the final stage, the policy's stability is limited compared to the baseline, as shown in Figure 12, which reaches a reward around 1500 by 8M timesteps and a reward of 2000 by 10M timesteps. In essence, depthwise expansion lags behind a fully trained, static network of the same final size.

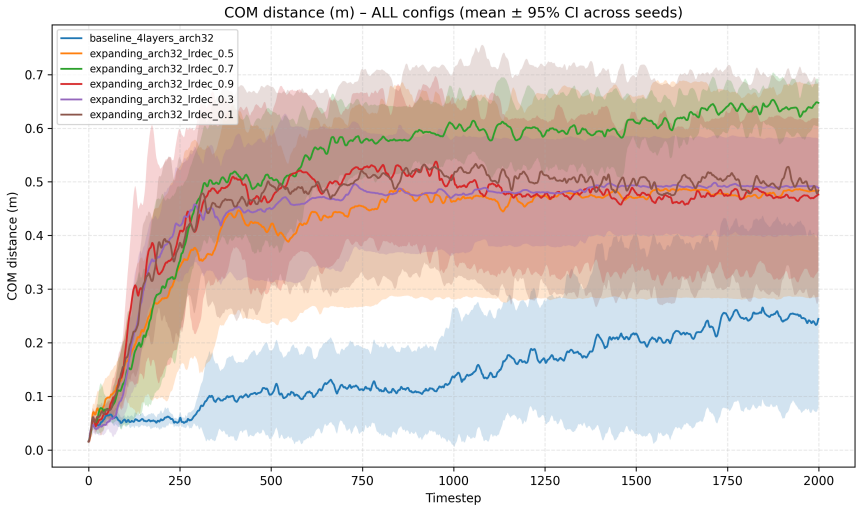

Figure 8: Comparison of the Average Distance between the COM and Center of Feet Per Timestep

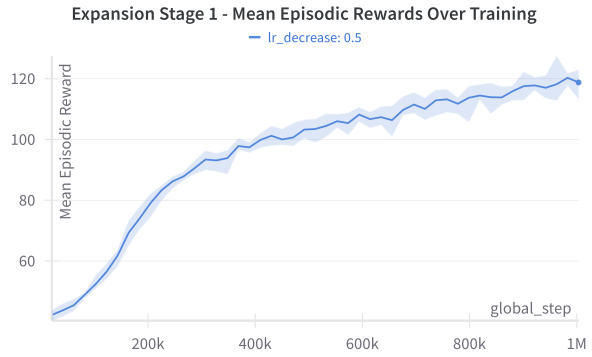

Figure 9: Mean Episodic Reward over the first expansion stage for the expanding network with a learning rate decay factor of 0.5. Averaged over 5 training seeds.

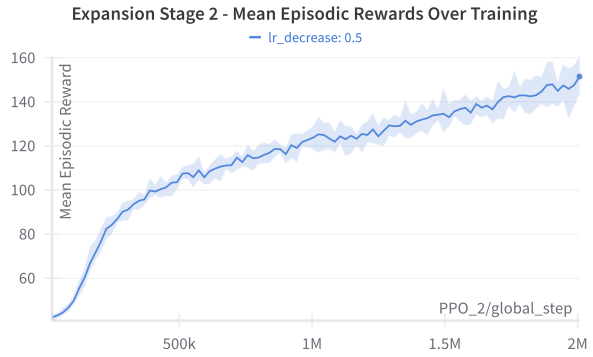

Figure 10: Mean Episodic Reward over the second expansion stage for the expanding network with a learning rate decay factor of 0.5. Averaged over 5 training seeds.

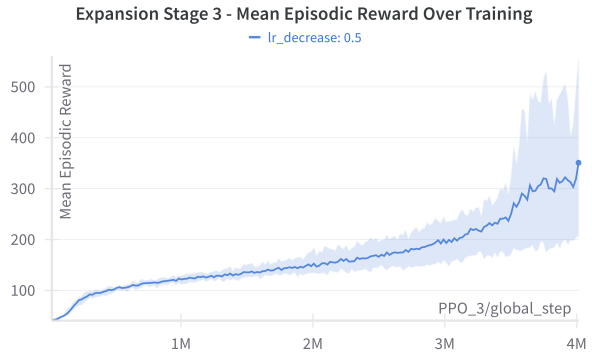

Figure 11: Mean Episodic Reward over the final expansion stage. Averaged over 5 training seeds.

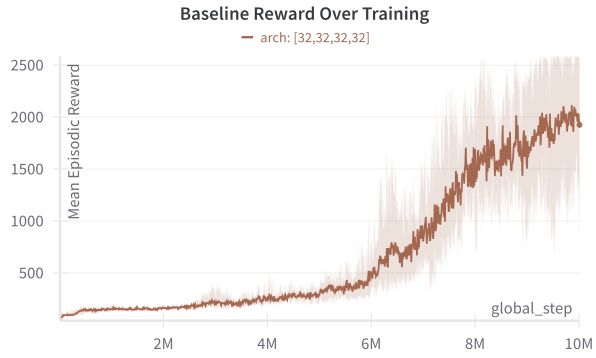

Figure 12: Mean Episodic Reward for the static baseline. Averaged over 5 training seeds.

5.5.2. (64, 64) → (64, 64, 64) → (64, 64, 64, 64) Expansion Path

Table 11 outlines similar overall trends in a comparison between a (64, 64, 64, 64) static baseline to expanding networks that grow from $(64, 64) \rightarrow (64, 64, 64) \rightarrow (64, 64, 64, 64)$. The baseline trained for 10 million timesteps attains the highest final reward of 1296, whereas all expanding variants remain below 500. The average distance between the COM and the center of the humanoid's feet is again noticeably smaller for the baseline (0.273m), indicating a more stable, upright posture. In comparison, the expansion COM displacements range from 0.396 to 0.546. Among the expansion, an LR-decay factor of 0.7 yields the highest final reward (410.33). However, an LR-decay factor of 0.1 yields the best COM stability. After qualitatively examining the accompanying videos, we

can attribute this to the fact that the former manages to stand for slightly longer but ends in a more sprawled out position, thus slightly increasing the reported COM metric but also increasing the reward. Overall, larger networks appeared more challenging to train, both for static and expanding architectures, as compared to Table 10.

Table 11: Comparison of $(64, 64) \rightarrow (64, 64, 64) \rightarrow (64, 64, 64, 64)$ sized expanding network to a static $(64, 64, 64, 64)$ sized baseline. All numbers are means over the five training seeds, upon each of which 5 evaluation episodes were run.

Config	LR-decay	Training Wall Time (hr)	Training Timesteps	COM* (m)	Energy** (J)	Final Evaluation Reward
Baseline	-	5.4	10M	0.273	18.49	**1296**
Expanding	0.1	3.1	[1M, 2M, 4M]	0.396	11.48	241.31
Expanding	0.3	3.13	[1M, 2M, 4M]	0.425	8.73	319.06
Expanding	0.5	3.15	[1M, 2M, 4M]	0.439	9.44	345.37
Expanding	0.7	3.13	[1M, 2M, 4M]	0.506	6.289	**410.33**
Expanding	0.9	3.15	[1M, 2M, 4M]	0.546	11.98	336.23

* Calculated as the average xy distance between the COM and the center of the humanoid's feet per evaluation timestep.

** Calculated as the average energy expenditure per evaluation timestep.

5.6. Widthwise Ablation Studies

5.6.1. (4, 4, 16) $\rightarrow$ (8, 8, 16) $\rightarrow$ (16, 16, 16) $\rightarrow$ (32, 32, 16) $\rightarrow$ (64, 64, 16) Expansion Path

Table 12 compares a static-width (64, 64, 16) baseline to expanding networks that grow from $(4, 4, 16)$ to $(64, 64, 16)$ over 4 expansions, doubling the width of the first two layers and the length of training for each phase. The baseline network achieves a high final reward of 1840, significantly outperforming all expanding configurations, whose final rewards range between 168 and 289. The baseline's COM displacement is also far lower at an average of 0.288, indicating better stability and less frequent falls, as compared in Figure 13. Among the expanding networks, a learning rate decay factor of 0.1 yields the highest final reward, while a learning rate decay factor of 0.9 yields the lowest.

Table 12: Comparison of $(4,4,16) \rightarrow (8,8,16) \rightarrow (16,16,16) \rightarrow (32,32,16) \rightarrow (64,64,16)$ expanding network to a static $(64,64,16)$ sized baseline. All numbers are means over the five training seeds, upon each of which 5 evaluation episodes were run.

Config	LR-decay	Training Wall Time (hr)	Training Timesteps	COM* (m)	Energy** (J)	Final Evaluation Reward
Baseline	–	6.83	10M	0.288	12.354	**1840.16**
Expanding	0.1	2.24	[.25M, .5M, 1M, 2M, 4M]	0.718	0.001	**288.894**
Expanding	0.3	2.28	[.25M, .5M, 1M, 2M, 4M]	0.659	0.002	173.243
Expanding	0.5	2.29	[.25M, .5M, 1M, 2M, 4M]	0.783	0.001	254.409
Expanding	0.7	2.34	[.25M, .5M, 1M, 2M, 4M]	0.770	0.001	225.222
Expanding	0.9	1.88	[.25M, .5M, 1M, 2M, 4M]	0.771	0.003	167.710

* Calculated as the average xy distance between the COM and the center of the humanoid's feet per evaluation timestep.

** Calculated as the average energy expenditure per evaluation timestep.

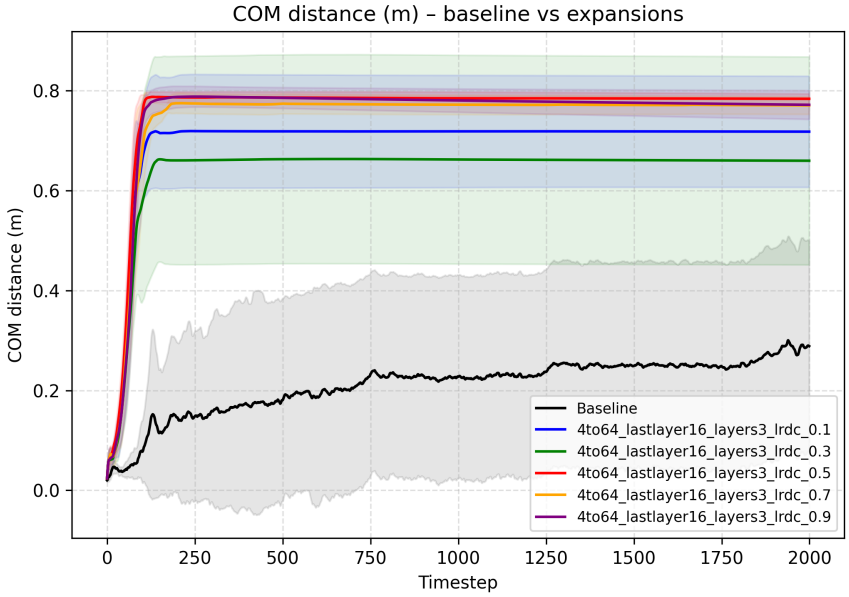

Figure 13: Comparison of the Average Distance between the COM and Center of Feet Per Timestep

5.6.2. (4, 4, 32) → (8, 8, 32) → (16, 16, 32) → (32, 32, 32) → (64, 64, 32) Expansion Path

In Table 13, the static baseline trained to 10M timesteps attains a final evaluation reward of 2327.47, far surpassing any of the expanding network's rewards, which range from 222 to 323. The baseline's COM displacement is just 0.117m on average. The expanding networks' COM displacements range between 0.612–0.763 m, and Figure 14 shows a clear spike early in evaluation, indicating falls.

Table 13: Comparison of $(4,4,32) \rightarrow (8,8,32) \rightarrow (16,16,32) \rightarrow (32,32,32) \rightarrow (64,64,32)$ expanding network to a static $(64,64,32)$ sized baseline. All numbers are means over the five training seeds, upon each of which 5 evaluation episodes were run.

Config	LR-decay	Training Wall Time (hr)	Training Timesteps	COM* (m)	Energy** (J)	Final Evaluation Reward
Baseline	–	5.45	10M	0.117	19.472	**2327.469**
Expanding	0.1	2.55	[1M, 2M, 4M]	0.750	0.000	249.929
Expanding	0.3	2.26	[1M, 2M, 4M]	0.732	0.007	**323.002**
Expanding	0.5	3.02	[1M, 2M, 4M]	0.763	0.001	222.486
Expanding	0.7	2.50	[1M, 2M, 4M]	0.612	0.003	232.997
Expanding	0.9	1.82	[1M, 2M, 4M]	0.759	0.003	276.131

* Calculated as the average xy distance between the COM and the center of the humanoid's feet per evaluation timestep.

** Calculated as the average energy expenditure per evaluation timestep.

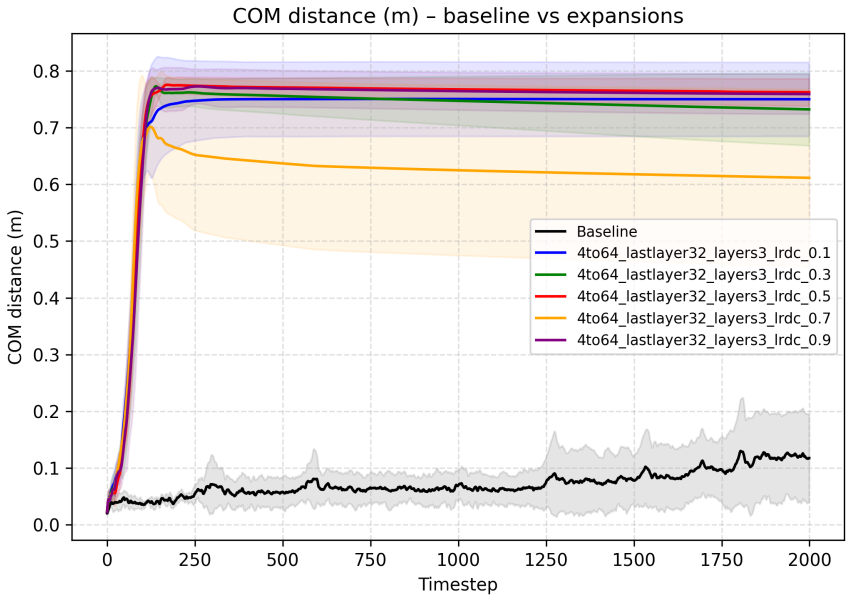

Figure 14: Comparison of the Average Distance between the COM and Center of Feet Per Timestep

5.6.3. (4, 4, 64) → (8, 8, 64) → (16, 16, 64) → (32, 32, 64) → (64, 64, 64) Expansion Path

Table 14 shows a similar trend. The static baseline achieves a reward of 1339.34, compared to 121.32–273.90 for the expanding runs. The baseline's average COM displacement is 0.408 m, reflecting moderate stability. All expanding variants exhibit higher displacement values (0.64–0.73m),

again suggestive of less stable movement and more falls, as plotted in Figure 15. The best reward among the expansions occurs at a learning rate decay of 0.7.

Table 14: Comparison of $(4,4,64) \rightarrow (8,8,64) \rightarrow (16,16,64) \rightarrow (32,32,64) \rightarrow (64,64,64)$ expanding network to a static $(64,64,64)$ sized baseline. All numbers are means over the five training seeds, upon each of which 5 evaluation episodes were run.

Config	LR-decay	Training Wall Time (hr)	Training Timesteps	COM* (m)	Energy** (J)	Final Evaluation Reward
Baseline	–	5.61	10M	0.408	13.077	**1339.338**
Expanding	0.1	1.95	[1M, 2M, 4M]	0.680	0.118	156.454
Expanding	0.3	2.13	[1M, 2M, 4M]	0.643	0.101	158.735
Expanding	0.5	1.88	[1M, 2M, 4M]	0.687	0.019	121.317
Expanding	0.7	2.23	[1M, 2M, 4M]	0.709	0.006	**273.904**
Expanding	0.9	2.95	[1M, 2M, 4M]	0.730	0.036	186.355

* Calculated as the average xy distance between the COM and the center of the humanoid's feet per evaluation timestep.

** Calculated as the average energy expenditure per evaluation timestep.

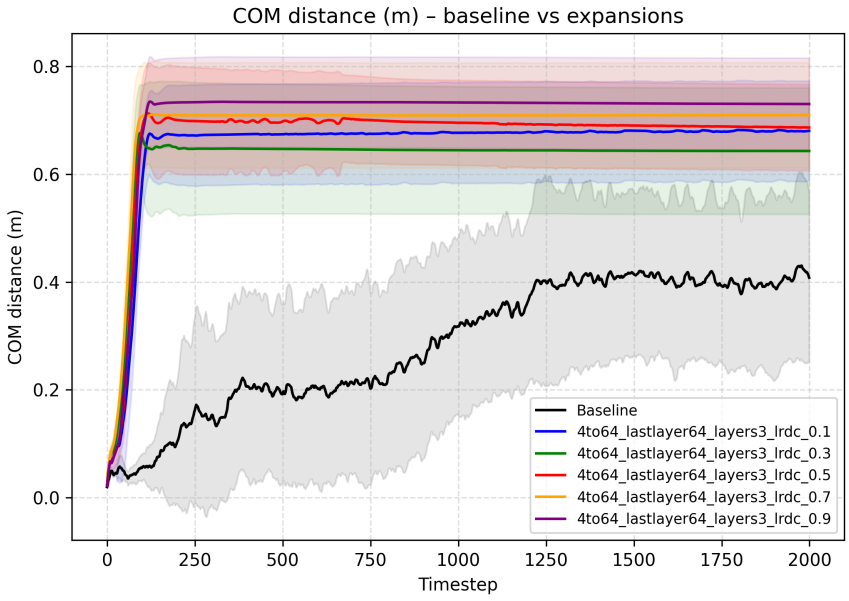

Figure 15: Comparison of the Average Distance between the COM and Center of Feet Per Timestep

5.6.4. (4, 4, 4, 16) → (8, 8, 8, 16) → (16, 16, 16, 16) → (32, 32, 32, 16) → (64, 64, 64, 16)

As shown in Table 15, the static baseline achieves a final reward of 772, while the best expanding

network achieves a final reward of 291 at a learning rate decay factor of 0.3. The baseline's average COM displacement is 0.445, as compared to ranges between 0.57 and 0.64 in the expanding networks. Figure 16 demonstrates that falls occur within the first 250 timesteps of evaluation episodes across expanding networks, while the baseline only significantly destabilizes between 500-1000 timesteps.

Table 15: Comparison of $(4,4,4,16) \rightarrow (8,8,8,16) \rightarrow (16,16,16,16) \rightarrow (32,32,32,16) \rightarrow (64,64,64,16)$ expanding network to a static $(64,64,64,16)$ sized baseline. All numbers are means over the five training seeds, upon each of which 5 evaluation episodes were run.

Config	LR-decay	Training Wall Time (hr)	Training Timesteps	COM* (m)	Energy** (J)	Final Evaluation Reward
Baseline	–	5.08	10M	0.445	5.787	**771.981**
Expanding	0.1	3.12	[1M, 2M, 4M]	0.568	0.004	163.599
Expanding	0.3	3.37	[1M, 2M, 4M]	0.633	0.011	**291.353**
Expanding	0.5	3.44	[1M, 2M, 4M]	0.598	0.007	262.198
Expanding	0.7	3.63	[1M, 2M, 4M]	0.637	0.001	148.454
Expanding	0.9	2.81	[1M, 2M, 4M]	0.608	0.023	180.662

* Calculated as the average xy distance between the COM and the center of the humanoid's feet per evaluation timestep.

** Calculated as the average energy expenditure per evaluation timestep.

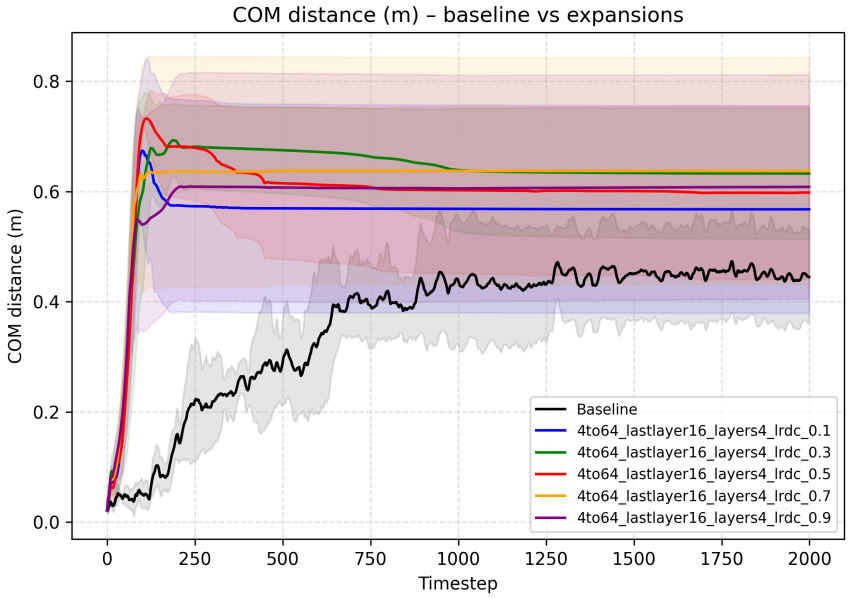

Figure 16: Comparison of the Average Distance between the COM and Center of Feet Per Timestep

5.6.5. (4, 4, 4, 32) → (8, 8, 8, 32) → (16, 16, 16, 32) → (32, 32, 32, 32) → (64, 64, 64, 32)

As shown in Table 16, the static baseline achieves a final reward of 1031, while the best expanding network achieves a final reward of 314 at a learning rate decay factor of 0.3. The baseline's average COM displacement is 0.461, as compared to ranges around 0.7 in the expanding networks. Figure 16 demonstrates that falls occur within the first 250 timesteps of the evaluation episodes across expanding networks, while the baseline only significantly destabilizes between 500-1000 timesteps.

Table 16: Comparison of $(4,4,4,32) \rightarrow (8,8,8,32) \rightarrow (16,16,16,32) \rightarrow (32,32,32,32) \rightarrow (64,64,64,32)$ expanding network to a static $(64,64,64,32)$ sized baseline. All numbers are means over the five training seeds, upon each of which 5 evaluation episodes were run.

Config	LR-decay	Training Wall Time (hr)	Training Timesteps	COM* (m)	Energy** (J)	Final Evaluation Reward
Baseline	–	4.81	10M	0.461	8.578	**1031.183**
Expanding	0.1	2.64	[1M, 2M, 4M]	0.723	0.002	302.862
Expanding	0.3	2.23	[1M, 2M, 4M]	0.688	0.047	**313.921**
Expanding	0.5	2.52	[1M, 2M, 4M]	0.735	0.003	243.381
Expanding	0.7	2.45	[1M, 2M, 4M]	0.766	0.003	186.607
Expanding	0.9	2.24	[1M, 2M, 4M]	0.734	0.001	271.845

* Calculated as the average xy distance between the COM and the center of the humanoid's feet per evaluation timestep.

** Calculated as the average energy expenditure per evaluation timestep.

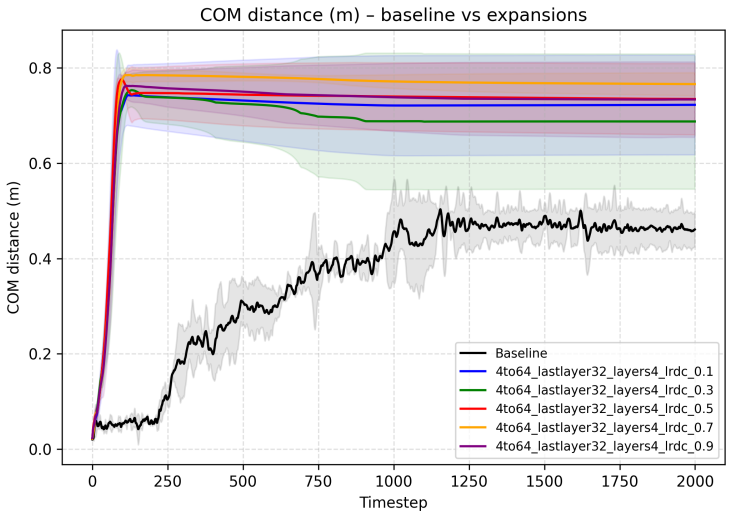

Figure 17: Comparison of the Average Distance between the COM and Center of Feet Per Timestep

5.6.6. (4, 4, 4, 64) → (8, 8, 8, 64) → (16, 16, 16, 64) → (32, 32, 32, 64) → (64, 64, 64, 64)

As shown in Table 17, the static baseline achieves a final reward of 1254, while the best expanding network achieves a final reward of 269 at a learning rate decay factor of 0.3. The baseline's average COM metric is 0.418, as compared to ranges closer to 0.7 in the expanding networks, as illustrated in Figure 18.

Table 17: Comparison of $(4,4,4,64) \rightarrow (8,8,8,64) \rightarrow (16,16,16,64) \rightarrow (32,32,32,64) \rightarrow (64,64,64,64)$ expanding network to a static $(64,64,64,64)$ sized baseline. All numbers are means over the five training seeds, upon each of which 5 evaluation episodes were run.

Config	LR-decay	Training Wall Time (hr)	Training Timesteps	COM* (m)	Energy** (J)	Final Evaluation Reward
Baseline	–	5.4	10M	0.418	12.080	**1254.165**
Expanding	0.1	2.31	[1M, 2M, 4M]	0.684	0.004	162.338
Expanding	0.3	1.86	[1M, 2M, 4M]	0.730	0.003	210.451
Expanding	0.5	1.85	[1M, 2M, 4M]	0.728	0.001	**268.919**
Expanding	0.7	2.27	[1M, 2M, 4M]	0.716	0.002	205.930
Expanding	0.9	1.88	[1M, 2M, 4M]	0.688	0.006	248.964

* Calculated as the average xy distance between the COM and the center of the humanoid's feet per evaluation timestep.

** Calculated as the average energy expenditure per evaluation timestep.

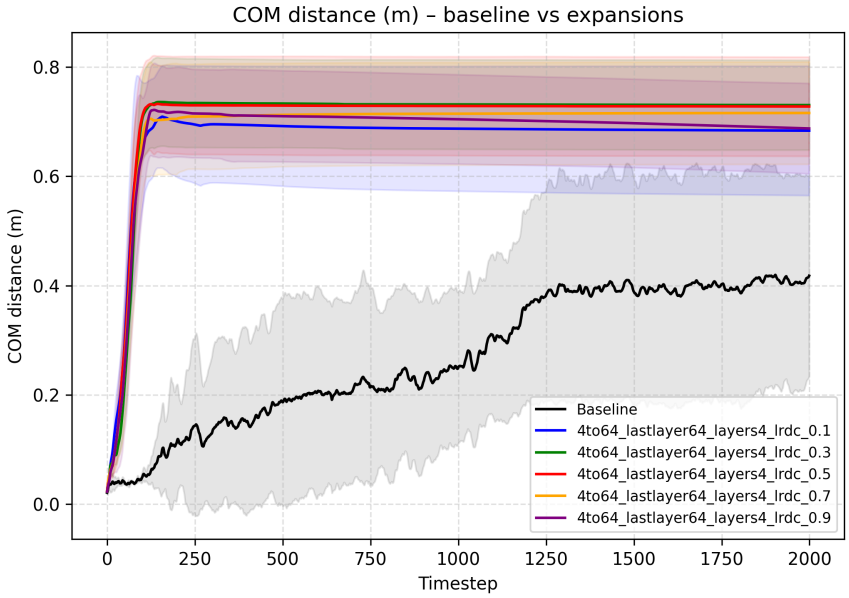

Figure 18: Comparison of the Average Distance between the COM and Center of Feet Per Timestep

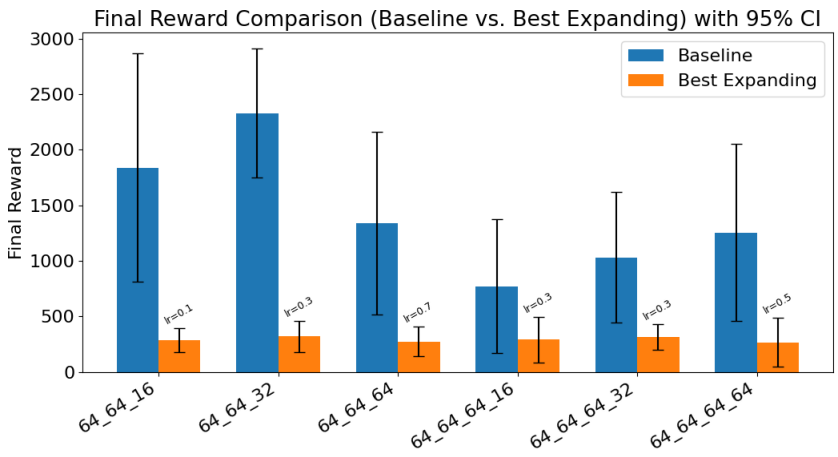

Figure 19: Final Rewards Across Width Expansions

5.6.7. Analysis Across Width Expansions

Across the board, the baseline networks uniformly achieve higher mean rewards but also exhibit wider confidence intervals than the best-expanding configurations, as shown in Figure 19. Meanwhile, the expanding approaches display consistently lower but more stable rewards, seldom exceeding the 300-400 range across all architectures.

When the network has four layers and a smaller last layer (16 instead of 64), the static baseline's advantage over the expanding approach narrows considerably. One explanation is that a smaller output dimension effectively compresses the action space, reducing the complexity of the final mapping from latent features to actions. In such a setting, incrementally adding width to earlier layers via the expanding architecture may serve as a beneficial inductive bias: the policy is forced to refine a lower-dimensional control strategy before scaling up its representational capacity. Unable to leverage that same inductive bias, the baseline performance drops in comparison. In contrast, when the last layer is large, the static baseline more readily exploits its capacity, leaving expanding networks lagging in final reward. When the network is limited to three layers, its smaller overall capacity makes the incremental benefits of expansion less apparent. In other words, if the baseline itself is not excessively large, it can still optimize effectively over 10 million steps without "overwhelming" capacity - making it harder for an expanding network to catch up. With four layers, however, the baseline's larger capacity can be harder to exploit immediately, while the expanding

network's staged growth helps prevent inefficient exploration in a high-dimensional parameter space. Consequently, that staged approach begins to pay off more noticeably in the deeper architecture.

The results also suggest that moderate decays (0.3, 0.5) help expanding networks the most. Hence, the synergy of four layers plus a moderate learning rate decay is where the expanding networks show their clearest advantage. While there remains a substantial gap in performance between the baselines and expanding networks, these results are still relatively impressive given that each expanding network only trains for [0.25M, 0.5M, 1M, 2M, 4M] timesteps on each task. Across all phases, this amounts to 8M timesteps in comparison to 10M for the baseline. Furthermore, at full network size, the expansions only train for 4M timesteps, which may not be quite enough time for larger sized networks to reach their full potential.

5.7. Comparing Depth vs Width Expansions

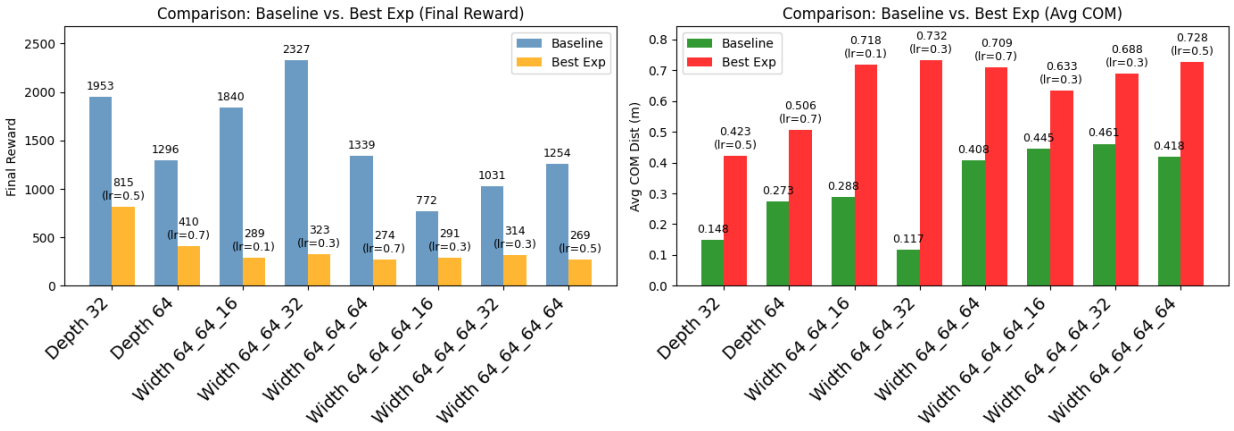

Figure 20: Comparison of Depth vs. Width Expansions

As shown in Figure 20, depth expansions achieve somewhat higher final rewards than width expansions do, considering only the best learning rate decay configuration for each. For instance, the Depth-32 expansion ultimately attains around 815 reward, whereas the width expansions typically top out below 400. Regarding the gap between baseline and expansion performance, the results are varied, with about half of the width expansions achieving a smaller difference compared to the corresponding baseline than the depth expansions. Indeed, 4 layer based expansions (whether

width or depth) tended to close the gap better than 3 layer width expansions in general, both in terms of final reward and average COM distance. However, in every case, the static baselines still outperform the expansions, suggesting much room for further refinement.

6. Conclusions and Pathways for Future Work

Our work explored two complementary strategies for humanoid locomotion: (1) a curriculum of subtasks aligned with developmental milestones and (2) expanding networks that incrementally grow in capacity, either in depth or width. Although neither approach fully solved the challenge of learning to walk, or even learning to get up, our experiments reveal key insights into how we might further refine and ultimately integrate these two strategies.

With respect to the curriculum, baseline policies for individual subtasks generally performed well under shorter training but sometimes regressed with extended training time, especially for the prone and supine positions. Incorporating RND mitigated this issue by enhancing exploration, dramatically improving learning stability. When subtasks were combined into a multi-stage curriculum, transferring knowledge across tasks proved challenging. A curriculum ordering of prone $\rightarrow$ crawl $\rightarrow$ knees $\rightarrow$ standing with boxes $\rightarrow$ standing showed the best transfer, resulting in lower training time intermediate subtask and higher final performance. In general, trials that transitioned between tasks earlier (either via more modest reward thresholds or lower time limits) performed better than trials with longer training on each stage, likely because the model did not overfit to previous tasks. In the future, we aim to more systematically tune reward thresholds and curriculum orderings, and we see significant promise in incorporating RND to enhance exploration across all tasks.

With respect to the expanding network, depthwise and widthwise expansions lagged behind static baselines trained at final capacity from the start. However, moderate learning-rate decay factors seemed to improve expansion performance, and a more extensive search of this hyperparameter space may help close this gap. We also plan to run future experiments with longer training times and perhaps explore varying the length of each phase. We suspect that 8 million timesteps may not be sufficient to evaluate full learning potential, and there may also be benefits in changing

the ratio of timesteps on each phase from a simple doubling formula. While we initially planned to evaluate energy expenditure alongside reward and COM displacement, the frequent early falls among expanding networks made this metric less meaningful, though they remain an important consideration for future work. Likewise, it is difficult to evaluate the wall clock savings of expanding networks, given that their lower performance inevitably lowered the mean episode length of training as compared to the more successful baselines. However, given that expanding networks train at smaller capacity in early phases, we expect to still see improvements in wall clock time in the future.

Finally, across all of our experiments, the size of the output layer significantly affected performance, as using a smaller size effectively compressed the action space and created an information bottleneck. While our width-wise expanding networks seemed better able to handle this challenge, we plan to scale up output layers in the future. In particular, this may help improve performance for the curriculum where our policy may need to be more expressive.

Ultimately, our chief limitation this semester was time. Because the Mujoco environments run on CPU and are the major bottleneck, training is enormously time consuming. In the near future, we plan to switch to Jax for faster, more scalable training, which will allow us to run more comprehensive testing with less effort. Nevertheless, our initial results indicate the potential synergy of uniting these methods into a single framework: an expanding network operating within a curriculum. By allocating capacity only when needed and avoiding overfitting to early tasks, such a system may better transfer knowledge across tasks while mitigating catastrophic forgetting. Further integrating RND for exploration and carefully selecting output-layer sizes could round out this approach, ultimately bringing us closer to emulating the efficient, structured learning process of human motor development.

7. Acknowledgments

Thank you very much to Tom Griffiths for providing a wonderful independent work experience. I am especially indebted to Dilip Arumugam and Gianluca Bencomo, whose generous mentorship taught me an enormous amount over the course of this semester. This project would not have been

possible without your guidance, and I look forward to continuing to build upon it together in the future.

8. Honor Code

This represents my own work in accordance with University regulations.

\s Natalia Espinosa Dice

References

[1] K. E. Adolph and S. R. Robinson, "The Road to Walking: What Learning to Walk Tells Us About Development," in *The Oxford Handbook of Developmental Psychology, Vol. 1: Body and Mind*, P. D. Zelazo, Ed. Oxford University Press, 2013, available at https://doi.org/10.1093/oxfordhb/9780199958450.013.0015.

[2] J. L. Elman, "Learning and Development in Neural Networks: The Importance of Starting Small," *Cognition*, vol. 48, no. 1, pp. 71–99, 1993, available at https://doi.org/10.1016/0010-0277(93)90058-4. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0010027793900584

[3] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming Catastrophic Forgetting in Neural Networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017, available at https://www.pnas.org/doi/10.1073/pnas.1611835114.

[4] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive Neural Networks," 2022, available at https://arxiv.org/abs/1606.04671.

[5] S. J. Bell and N. D. Lawrence, "The Effect of Task Ordering in Continual Learning," 2022, available at https://arxiv.org/abs/2205.13323.

[6] K. E. Adolph and J. M. Franchak, "The Development of Motor Behavior," *Wiley Interdisciplinary Reviews: Cognitive Science*, vol. 8, no. 1-2, p. e1430, 2017, available at https://pmc.ncbi.nlm.nih.gov/articles/PMC5182199/.

[7] L. Bao, J. Humphreys, T. Peng, and C. Zhou, "Deep Reinforcement Learning for Bipedal Locomotion: A Brief Survey," 2025, available at https://arxiv.org/abs/2404.17070.

[8] Y. Yang, T. Zhang, E. Coumans, J. Tan, and B. Boots, "Fast and Efficient Locomotion via Learned Gait Transitions," *CoRR*, vol. abs/2104.04644, 2021, available at https://arxiv.org/abs/2104.04644.

[9] R. P. Singh, M. Benallegue, M. Morisawa, R. Cisneros, and F. Kanehiro, "Learning Bipedal Walking On Planned Footsteps For Humanoid Robots," 2022, available at https://arxiv.org/abs/2207.12644.

[10] J. Andreas, D. Klein, and S. Levine, "Modular Multitask Reinforcement Learning with Policy Sketches," 2017, available at https://arxiv.org/abs/1611.01796.

[11] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Reinforcement Learning for Versatile, Dynamic, and Robust Bipedal Locomotion Control," 2024, available at https://arxiv.org/abs/2401.16889.

[12] P. Xu, X. Shang, V. Zordan, and I. Karamouzas, "Composite Motion Learning with Task Control," *ACM Transactions on Graphics*, vol. 42, no. 4, p. 1–16, 2023, avaialble at url{http://dx.doi.org/10.1145/3592447.

[13] X. B. Peng, Y. Guo, L. Halper, S. Levine, and S. Fidler, "ASE: Large-scale Reusable Adversarial Skill Embeddings for Physically Simulated Characters," *ACM Transactions on Graphics*, vol. 41, no. 4, 2022, available at http://dx.doi.org/10.1145/3528223.3530110.

[14] S. Ha, P. Xu, Z. Tan, S. Levine, and J. Tan, "Learning to Walk in the Real World with Minimal Human Effort," *CoRR*, 2020, available at https://arxiv.org/abs/2002.08550.

[15] T. Tao, M. Wilson, R. Gou, and M. van de Panne, "Learning to Get Up," 2022, available at https://arxiv.org/abs/2205.00307.

[16] Z. Fu, A. Kumar, J. Malik, and D. Pathak, "Minimizing Energy Consumption Leads to the Emergence of Gaits in Legged Robots," *CoRR*, 2021, available at https://arxiv.org/abs/2111.01674.

[17] W. Yu, G. Turk, and C. Liu, "Learning Symmetry and Low-energy Locomotion," *CoRR*, vol. abs/1801.08093, 2018, available at http://arxiv.org/abs/1801.08093.

[18] F. Abdolhosseini, H. Y. Ling, Z. Xie, X. B. Peng, and M. van de Panne, "On Learning Symmetric Locomotion," in *ACM SIGGRAPH Conference on Motion, Interaction and Games (MIG)*, 2019, available at https://doi.org/10.1145/3359566.3360070.

[19] K. Wang, I. Javali, M. Bortkiewicz, T. Trzciński, and B. Eysenbach, "1000 Layer Networks for Self-Supervised RL: Scaling Depth Can Enable New Goal-Reaching Capabilities," 2025, available at https://arxiv.org/abs/2503.14858.

[20] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," 2017, available at https://arxiv.org/abs/1707.06347.

[21] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," 2016.

[22] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-Baselines3: Reliable Reinforcement Learning Implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021, available at http://jmlr.org/papers/v22/20-1364.html.

[23] L. Biewald, "Experiment Tracking with Weights and Biases," 2020, software available from wandb.com. [Online]. Available: https://www.wandb.com/

[24] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[25] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by Random Network Distillation," 2018, available at https://arxiv.org/abs/1810.12894.

A. Link to Repository

Code can be found at:
`https://github.com/gianlucabencomo/Learning-to-Walk-Like-Humans-Do`.
Weights and Biases Logging can be found at:
`https://wandb.ai/ne3496-princeton-university/L2W-4-4-25`.

B. Additional Width Wise Expansion Results

Table 18: Comparison of $(8,8,64) \rightarrow (16,16,64) \rightarrow (32,32,64) \rightarrow (64,64,64) \rightarrow (128,128,64)$ expanding network to a static $(128,128,64)$ sized baseline. All numbers are means over the five training seeds, upon each of which 5 evaluation episodes were run.

Config	LR-decay	Training Wall Time (hr)	Training Timesteps	COM* (m)	Energy** (J)	Final Evaluation Reward
Baseline	–	9.7	10M	0.226	15.182	1834.370
Expanding	0.1	2.85	[1M, 2M, 4M]	0.794	0.042	259.356
Expanding	0.3	2.40	[1M, 2M, 4M]	0.697	0.008	222.134
Expanding	0.5	2.88	[1M, 2M, 4M]	0.689	0.003	188.181
Expanding	0.7	2.61	[1M, 2M, 4M]	0.657	0.001	150.437
Expanding	0.9	2.64	[1M, 2M, 4M]	0.755	0.001	239.698

* Calculated as the average xy distance between the COM and the center of the humanoid's feet per evaluation timestep.
** Calculated as the average energy expenditure per evaluation timestep.

Table 19: Comparison of $(8,8,128) \rightarrow (16,16,128) \rightarrow (32,32,128) \rightarrow (64,64,128) \rightarrow (128,128,128)$ expanding network to a static $(128,128,128)$ sized baseline. All numbers are means over the five training seeds, upon each of which 5 evaluation episodes were run.

Config	LR-decay	Training Wall Time (hr)	Training Timesteps	COM* (m)	Energy** (J)	Final Evaluation Reward
Baseline	5.0	4.82	10M	0.376	12.777	1089.028
Expanding	0.1	3.27	[1M, 2M, 4M]	0.623	0.032	180.596
Expanding	0.3	2.93	[1M, 2M, 4M]	0.614	0.005	113.558
Expanding	0.5	2.62	[1M, 2M, 4M]	0.691	0.100	213.882
Expanding	0.7	2.60	[1M, 2M, 4M]	0.658	0.180	57.885
Expanding	0.9	2.78	[1M, 2M, 4M]	0.684	0.029	180.583

* Calculated as the average xy distance between the COM and the center of the humanoid's feet per evaluation timestep.
** Calculated as the average energy expenditure per evaluation timestep.

Table 20: Comparison of $(8,8,8,64) \rightarrow (16,16,16,64) \rightarrow (32,32,32,64) \rightarrow (64,64,64,64) \rightarrow (128,128,128,64)$ expanding network to a static $(128,128,128,64)$ sized baseline. All numbers are means over the five training seeds, upon each of which 5 evaluation episodes were run.

Config	LR-decay	Training Wall Time (hr)	Training Timesteps	COM* (m)	Energy** (J)	Final Evaluation Reward
Baseline	–	11.35	10M	0.428	6.262	1495.965
Expanding	0.1	3.08	[1M, 2M, 4M]	0.727	0.352	193.408
Expanding	0.3	3.55	[1M, 2M, 4M]	0.740	0.014	143.805
Expanding	0.5	3.21	[1M, 2M, 4M]	0.769	0.024	175.779
Expanding	0.7	3.57	[1M, 2M, 4M]	0.733	0.110	274.547
Expanding	0.9	3.37	[1M, 2M, 4M]	0.641	0.141	171.168

* Calculated as the average xy distance between the COM and the center of the humanoid's feet per evaluation timestep.
** Calculated as the average energy expenditure per evaluation timestep.

Table 21: Comparison of $(8,8,8,128) \rightarrow (16,16,16,128) \rightarrow (32,32,32,128) \rightarrow (64,64,64,128) \rightarrow (128,128,128,128)$ expanding network to a static $(128,128,128,128)$ sized baseline. All numbers are means over the five training seeds, upon each of which 5 evaluation episodes were run.

Config	LR-decay	Training Wall Time (hr)	Training Timesteps	COM* (m)	Energy** (J)	Final Evaluation Reward
Baseline	–	4.96	10M	0.362	7.458	716.952
Expanding	0.1	2.56	[1M, 2M, 4M]	0.769	0.009	196.637
Expanding	0.3	2.56	[1M, 2M, 4M]	0.721	0.001	155.220
Expanding	0.5	2.63	[1M, 2M, 4M]	0.786	0.016	141.568
Expanding	0.7	2.61	[1M, 2M, 4M]	0.792	0.001	129.266
Expanding	0.9	2.55	[1M, 2M, 4M]	0.726	0.001	285.274

* Calculated as the average xy distance between the COM and the center of the humanoid's feet per evaluation timestep.
** Calculated as the average energy expenditure per evaluation timestep.