Maintaining Plasticity in Dynamic Stationary Environments via Continual Reinforcement Learning

Jack Michaels [*1] **Sathya Edamadaka** [*1]

GitHub: https://github.com/jmichaels32/CS234

Note: This idea was suggested by Saurabh Kumar through the class project idea submission form. We credit him for his insights and leadership here.

Abstract

The ability of a deep learning (DL) model to generalize results outside of train, validation and test datasets is essential for inference once deployed. Classic reinforcement learning (RL) methods are often trained with a specific environment in mind, resulting in limited adaptability to new conditions. This out of distribution performance becomes especially important to consider when designing systems that require robustness for the sake of public safety. To address this, this paper applies state-of-the-art architectural modifications to policy gradient methods and assess their ability to generalize within a discretely changing MuJoCo environment. As a baseline, we use raw proximal policy optimization (PPO) with a vanilla multi-layer perceptron (MLP) as a policy predictor. From there, we modify PPO by incorporating L2 init and concatenated ReLU generalization techniques from the literature. We see strong performance across the changing environmental parameters for L2 init with clear performance improvements over baseline during repeated and novel environment exposure. In addition, we see performance enhancements caused by concatenated ReLU when modifying specific environmental parameters, though these improvements are highly dependent on the parameter at play.

[*]Equal contribution [1]Department of Computer Science, Stanford University. Correspondence to: Jack Michaels <jackfm@stanford.edu>, Sathya Edamadaka <sath@stanford.edu>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

1. Introduction

Plasticity refers to the ability of an AI agent (in particular, those that use neural networks, in reference to neuroplasticity) to adapt to new situations or learn new skills over time (Lyle et al., 2023). When deploying RL agents into the real world, though an agent might have encountered a diverse set of environments during training, it may be presented with completely novel situations or ever changing environments. While classic RL methods assume a stationary environment, as we progress the applicability of RL we begin to consider domains with higher risk and changing environments (Padakandla, 2021). These environments can be continually changing where one parameter slowly, and continually, transitions from one unique value to another or they can be discretely changing. In this paper, we consider the discrete case where the environment rapidly transitions to another unique value as opposed to the continual case. This choice was made due to the documented performance collapse associated with continual environmental changes which are often caused by the model lacking sufficient training time within any particular environmental setting (Abbas et al., 2023). Though continuity is often seen more in the real world, we argue that rapid, discrete changes in the environment are just as common and often justifiably require better performance.

Depending on the vitality of this agent, performance in these unseen environments is not only paramount, but necessary. Examples of dangerous environments that demand performance where decisions are irreversible include autonomous driving, financial trading, or nearly all applications related to medical diagnosis and treatment (Padakandla, 2021). These environments often have such diverse environmental parameters that it is impractical or impossible to perfectly model them during training meaning plasticity is a required feature of trained models for robust performance during test time (Buesing et al., 2018).

Solutions to this plasticity problem are highly non-trivial; during training time we need to both train an agent to perform on the environment it is currently being presented and, at some arbitrary higher level, have the agent understand how it achieved said performance so it can quickly transfer its learning to novel environments by maintaining plasticity.

This task draws similarities to the general AI task of regularization since a relatively straightforward solution would be a well regularized algorithm since it assumes so little about its current training schedule and thus can achieve high average performance across all environmental conditions (Kumar et al., 2023). However, there is an opportunity for negligence here since we may optimize for average accuracy across all environments instead of the ability for an agent to accurately transfer learning to new environments. Such a difference in agents is the difference between a meta-learned or continual backprop objective versus a regularized objective (Dohare et al., 2021) (Finn et al., 2017). This delineation is quite subtle but it is the difference between a truly intelligent algorithm and an accidentally passable algorithm.

We approached this problem through the lens of regularization. We incorporate L2 initialization and concatenated ReLU, both forms of reinforcement learning regularization, into a naive PPO algorithm with an MLP policy predictor. At the price of granularity on particular environments (Kumar et al., 2023), we make these architectural modifications to increase the generality of our system during test time.

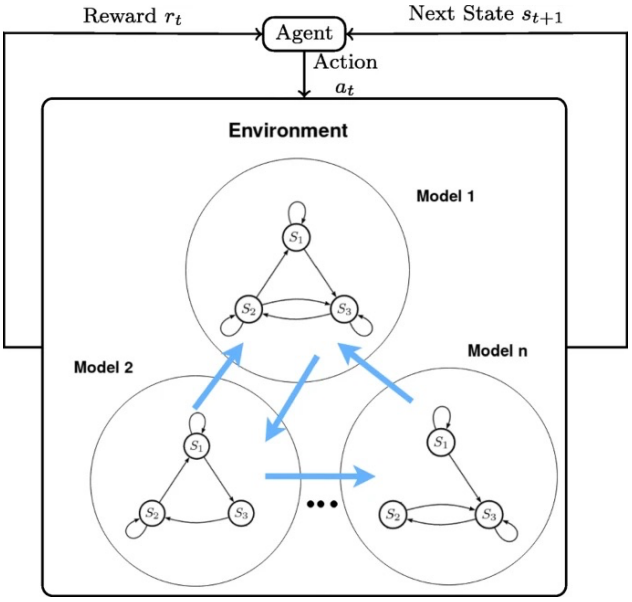

Figure 1. Context Q-learning uses stored previous experience to continually train on the same environments instead of resetting previous progress when a new environment is encountered. It does so through a detection mechanism injected into its modified Q-learning algorithm.

2. Related Works

Remarkably performative RL systems have been created in recent years for stationary policies given the increasing robustness of current architectures (Lillicrap et al., 2015) (Silver et al., 2017). These stationary environments may achieve reliable success within their distribution of training environments, yet once a novel situation is presented to them their success drastically diminishes. Further architectural modifications are required to create any lasting performance across the vast possible environments a model may be presented.

2.1. Techniques Involving Regularization

Common regularization techniques such as layer norm, dropout, or L2 regularization are often first steps when considering regulariation type solutions to plasticity. While these solutions are marginally effective, other solutions such as L2 initialization consistently achieve performance in non-stationary settings (Kumar et al., 2023). This method modifies the typical L2 regularization where instead of penalizing large weights we penalize large deviations from initial weight values, acting as a sort of reinitialization such as a continual learning acts.

Adjacent to L2 initialization is concatenated ReLU which preserves both the positive and negative phase information of ReLU (Shang et al., 2016). This method operates under the guise that under inspection of CNN filters there appear to be learned filters with opposite phases. Therefore, architectural optimizations can be made where we effectively consolidate these redundant layers using concatenated ReLU which produces a more robust model, allowing us to handle more parameters and thus more environments.

2.2. Continual Learning Techniques

Continual learning is an algorithmic approach to combating plasticity by learning in new environments using continual backprop (Dohare et al., 2021). Continual backprop leverages selective injection of random features into the learned function according to the utility of particular features with respect to performance, allowing the model to learn using the benefits of a semi-newly initialized model but remember using the previously kept features. Continual backprop's progressive feature reset allows the model to learn across environments and tasks, creating some form of ensemble of experts to ultimately maintain plasticity.

Context Q-learning is a continual learning algorithm that takes advantage of the knowledge that the environment may change by purposefully detecting such environment changes (Padakandla et al., 2020). This method modifies Q-learning to learn optimal policies for these different environments, keeping track of which environments have been seen and avoiding catastrophic forgetting by using a experience tuples. See Figure 2.2 for details on the non-stationary RL framework used by context Q-learning.

2.3. Meta-Learning Adjacent Techniques

To achieve the deeper understanding associated with learning to learn, the literature details various meta-learning adjacent methods for achieving plasticity.

In a general sense, meta learning (MAML) itself can be applied to changing environments (Finn et al., 2017). By assuming each environmental condition as its own task, we can apply meta-optimization to directly train the ability of a model to learn, especially when confronted with new environmental parameters. There has been some success here where RL methods trained with meta-reinforcement learning experience marginal increases in performance on novel environment exposures (Clavera et al., 2018), but this otherwise has experienced a detrimental side effect of attempting to learn under continually changing conditions referred to as catastrophic forgetting (Padakandla, 2021). Catastrophic forgetting is the degenerate case of training a model to be more plastic where no learning in general gets completed because the model is unable to remain in any given environmental parameter space for long enough to learn something substantial before it is presented with a new environmental parameter space.

Another approach adjacent to meta learning leverages an ensemble of selves to intrinsically handle different aspects of the changing environment (Dulberg et al., 2023). This approach details an emergent capability for ensemble of selves to explore and become robust to changing environmental parameters. This robustness may be attributed to the ability of the ensemble to reach some form of conflicting objective amongst one another, allowing each head to gather insights that others may not have assumed.

3. Approach

We studied how applying recent performant advancements in improving artificial neural network plasticity to a particular reinforcement learning problems could improve standard RL algorithms' average returns.

3.1. Simulation environment

We chose the Multi-Joint Dynamics with Contact environment (MuJoCo) as a reinforcement learning setting (Todorov et al., 2012). In particular, we set up a Hopper-v4 environment, which consists of four segments that connect between each other to form a leg (the first three form the upright portion of the leg and the last is a foot). An agent decides what torques to apply to the thigh rotor (first joint), leg rotor (second joint), and foot rotor (last, ankle joint). The reward provided by the environment is a total of a "healthy," "forward," and "control_cost" reward. The first rewards the agent for keeping the leg upright and from achieving too extreme of an angle at any joint (i.e. winding up too much). The second rewards the agent for moving the leg forward via hopping (hence the name of the environment), measured by the change in x position after taking an action. The third and last rewards the agent if it takes smaller actions (i.e. not springing forward with as much force as possible), thus encouraging the AI agent to stay within a reasonable torque space and not reach the potentially unrecoverable parameter space of large-magnitude torque actions.

We chose to change aspects of this environment to produce a changing set of conditions for RL models to adapt to over time. In particular, we chose four changing parameters: geometric friction, geometric margin, body mass, and anti-gravity components for the body.

Geometric friction controls the friction in the joint between each pair of leg pieces defined within the geometry of the Hopper apparatus. Geometric margin controls within what threshold about each joint and leg collisions and actions can occurr (i.e. preventing hopper segments from overlapping). Body mass controls the mass of all Hooper components, directly affecting the mobility of the device against gravity. Lastly, anti-gravity components reflect an anti-gravitational force that affects all aspects of the system.

3.2. Environment Change Agendas

3.2.1. Nonstationary Distributions

The exact schedule that parameter changes follow has an extreme impact on how successful existing RL models can be in environments that undergo them (Abel et al., 2024). In just the four parameters above, we could generate several different types of parameter changes. We segment them into two main groups: stationary dynamic processes and non-stationary ones. The difference lies in the distributions that are sampled when pulling data for upcoming examples. For the former, even if the parameter changes over time, if the distribution of values does not, the process remains stationary. Simple examples of this could be an impulse train or sinusoidal curve, which are periodic and thus stationary in that the distribution of data that yields datapoints separated by the wavelength are constant. On the other hand, data that has a (for example) moving mean as time goes on are non-stationary, in that their distributions are changing over time. As an example, friction in a joint that linearly increases (regardless of how slowly) over time is non-stationary because there exists no sliding window of length smaller than the total simulation length in which the friction means of each window across the entire agenda do not change.

We chose to demonstrate the effect parameter change agendas can have on model performance by training our baseline model (discussed in the next section) on three different parameter perturbation schedules. We varied geometric fric-

tion in three ways: linearly increasing, step-wise increasing with small increments (but the same maximum friction at the end of the training time), and step-wise increasing with large increments. We then observed the average returns throughout the training process to understand if the model is able to train on a nonstationary distribution, shown in Figure 2.

3.2.2. Stationary Distributions

On the other hand, using a stationary distribution could allow the RL agents to learn more easily. Therefore, we propose to separate training into 10 epochs (for a total of 3000000 training steps) and to perturb parameters according to an impulse train for the first 8 (shown in the right of Figures 5 and 6). For the 9th epoch, we test the RL agent by perturbing the parameter to a value not yet seen before, but one within the distribution of values seen in the first 8 epochs. Lastly, in the 10th epoch, the RL agent is tested on a completely out of distribution value from the ones observed in the first 9 epochs. This parameter change agenda affords us the ability to not only train on a stationary distribution, but also test on values of the parameter in and out of the distribution of values trained upon, both evaluating the performance of stationary distribution and how RL agents can perform on in-distribution and out-of-distribution values. A parameter change agenda following this pattern was generated for the geometric friction, mass value, geometric margin, and anti-gravity components. For the first two, we test a parameter change agenda with a smaller range of values, as well as an out of distribution value that isn't more than 2 times the maximum value seen in training (Fig. 5). For the second two parameters, we repeat the same process but try out-of-distribution values on the 10th epoch that are much larger than that of the first two parameter change agendas. How we've constructed the environment change agendas is one of the novel aspects of our work.

3.3. Baseline

We chose Proximal Policy Optimization (PPO) for our RL agent algorithm across all baseline and plastic solutions. PPO is an extremely well established RL algorithm with several published baselines and trusted implementations. We chose to opt for the clipped surrogate objective formulation in the PPO loss to ensure that policy updates are not too large, keeping the policy in a better parameter space (Schulman et al., 2017). We chose a simple, feed-forward linear layer multi-layered perceptron (MLP) network for the state-action space representation in both the actor and critic models (for the value function and policy parameters). Specifically, we had 64 nodes across 2 hidden layers with ReLU activation functions between.

In addition, it's important to understand if the new environmental parameter values after their perturbations (as per the agendas) push the RL problem into a regime that is unlearnable (i.e. a friction level that requires higher torques than what are rewarded in the control cost reward component). As a result, we also trained a baseline PPO model from scratch for one epoch at each unique parameter value (in addition to the first epoch, it was trained on the second, ninth, and tenth).

Environments were changed either continually or in steps (as per the shape of an impulse train or stepwise increasing function, shown in Figures 2, 5, and 6).

3.4. Plastic Solutions

Crucially, the plasticity of our PPO implementation solely relies on the MLP network. This is because the ability of an agent to quickly adapt to new information relies in how the state-value space embeddings, learned by the actor and critic neural network models (value function and policy network), can learn how to adapt its weighting of different rewards in a changing environment. In particular, neural networks are known to lose plasticity when processing moving data streams or performing inference on out of stationary distribution inputs. We chose to modify aspects of its architecture in order to improve its ability to continue learning in a changing, yet stationary, environment.

3.4.1. Regenerative Regularization

One method that has shown tremendous promise in improving the continual learning capabilities of neural networks is regenerative regularization (Kumar et al., 2023). In particular, Kumar et al. developed a method called L2 Init, a simple addition to the loss function that regularizes a model's parameters against the initial values they had before training. The authors showed that their approach, with the core insight that recent losses that should be insensitive to particular parameters drift towards their initial values, consistently mitigated plasticity loss compared to previously proposed approaches on several different types of problems. They formulated L2 Init as shown below (Kumar et al., 2023).

$$\mathcal{L}_{reg}(\theta) = \mathcal{L}_{train}(\theta) + \lambda||\theta - \theta_0||_2^2 \tag{1}$$

Given our impulse train parameter change agenda, we are specifically looking to train our policy MLPs on a set distribution (hence the first 8 epochs being one of two parameter values). Therefore, instead of choosing a completely random initial starting point to regularize to, which instead would be appropriate had we been trying to develop a truly general policy network performant for any range of parameter values, we chose to let the model train for one epoch without any L2 init loss, and then add in the term, setting θ_0 to be the model's parameters after one epoch. This not

only captures an initial, valid point in parameter space for the regularization loss to bias towards, but also still captures a set of parameters early enough in the training process to allow for sufficient local exploration to gain performance throughout the remaining epochs. Tuning the hyperparameter between values of 10^{-3} and 10^{2} could produce different results given the magnitude of the loss function.

3.4.2. Concatenated ReLU

Another promising method that can be quickly integrated into neural networks to improve their plasticity is changing nonlinearity activation functions from ReLUs to concatenated ReLUs (CReLU) (Shang et al., 2016). Its formulation is as follows.

$$f(x) = \textbf{CONCAT}[\textbf{ReLU}(x), \textbf{ReLU}(-x)] \qquad (2)$$

It was originally proposed for use after convolutional neural network and max pooling layers. Its rationale is to allow a neural network layer to be activated in both a positive and negative direction while maintaining the same degree of non-saturated non-linearity. This works because CReLU has an information preservation nature, as it conserves both negative and positive linear responses after neural network processing. This gives the previous neural network a reconstructive power when equipped with CReLU. Shang et al. showed that even though the input dimension of the next layer doubles, generalizability and improved performance under dynamic environments are achieved even with halving the output dimension to preserve the same total number of parameters in a layer.

3.4.3. Hybrid Approaches

In addition, since L2 Init and CReLU modify different parts of the PPO MLPs, it's entirely possible to include both of them at once. As a result, we decided to investigate how a hybrid approach between both of them could perform.

4. Results and Discussion

First we discuss how PPO performed when trained on dynamic, non-stationary environments, shown in Figure 2-4.

When changing geometric friction over the course of the simulation, we observe that in all cases, PPO struggles to learn to achieve high average returns. For the constant linear increase case (Fig. 2), with the parameter change agenda shown in the upper right, average returns initially drops off before increasing rapidly, showing that the PPO model begins to adapt to small, yet constant, changes in the environment. However, the continual increase in friction proves to be too much for the model to continue to adapt to as training continues, implying that the parameters had reached a point at which they could not return to previous positions in parameter space. Already, the need for biasing parameters back to the space they were in is needed, which L2 Init directly addresses.

Next, we observe in Figure 3 that geometric friction increasing in discrete, smaller steps also produces a similar return learning effect. After an initial peak and dropoff in average returns, the model spikes up around the 20th training episode before slowly failing to learn any new information about the changing environment. This indicates that a discretization in a smooth parameter change may not have an effect on the PPO model learning if the distribution of the parameter change is non-stationary.

Lastly, we test the extreme of this idea in Figure 4. Here, we test changing the friction much more sharply, but also allowing the model more time to adjust and learn before the next parameter change. Here, we observe a more reasonable result, with the model learning much more during each parameter regime, even despite large performance drops upon the sharp changes in the environment. This validates our choice to opt for a longer time scale parameter changing agenda with the impulse train, where we switch only after each entire epoch to a new parameter value.

We now turn our attention to each of the parameters we provided an environment change agenda. Each return plot for each model shows the spread of each model over 2 random seeds. To start, in Figure 5, we see how a variety of models perform when changing geometric friction and body mass with relatively small variation cases. Specifically, at the end of each impulse train, the last value of both parameters is only twice as large as the maximum value from the first 8 epochs. For geometric friction, we observe that PPO with L2 init regularization and $\lambda = 10^{-2}$ performs the best on each period of low parameter perturbation, and all of the models struggle to learn at all on the epochs with high geometric friction. We do observe, though, that starting training in a low friction epoch does provide a better starting point for PPO to train than just starting training from scratch in the high-friction epoch. We also observe a slow improvement between each high friction epoch, with slight increases each time the friction switches high in the impulse train. We observe that for the in-distribution but new friction value (9th epoch), the L2 init still outperforms all other models, although like the other low-friction epochs, CReLU and the other L2 init weighting are close behind. All models seem to struggle to learn on the last epoch, where a completely unseen value for friction is learned upon. We see that PPO retrained from scratch each epoch underperforms on the first inference value (9th epoch), but outperforms almost every model throughout the second inference value (10th epoch), which indicates that it may have been too high of a value to learn on in the first place. However, right near the end,

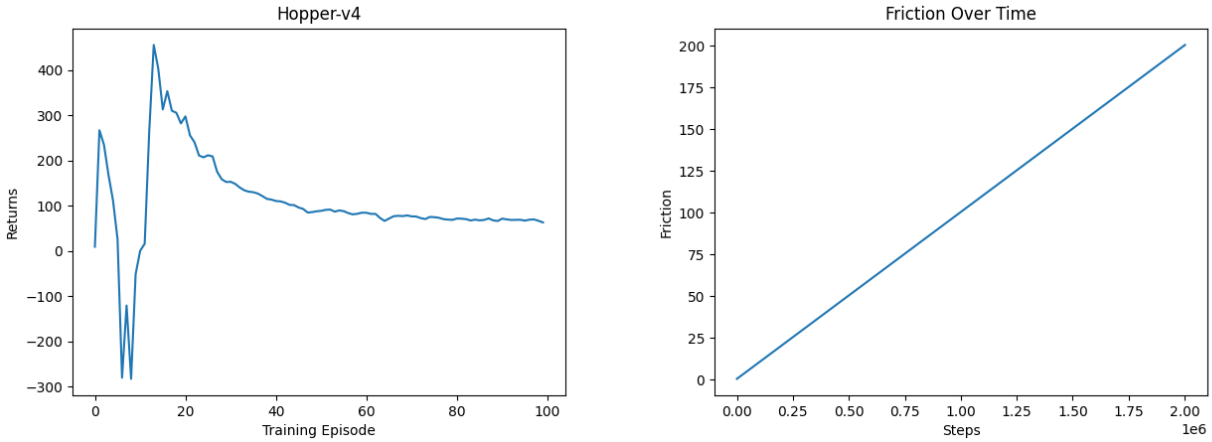

Figure 2. **Early environment change dynamics tests and their associated performance—linear increasing geometric friction:** Initial environment changing dynamics involved the continual modification of a particular parameter (in this particular case friction). Training episode and steps are different labeling units for the same process.

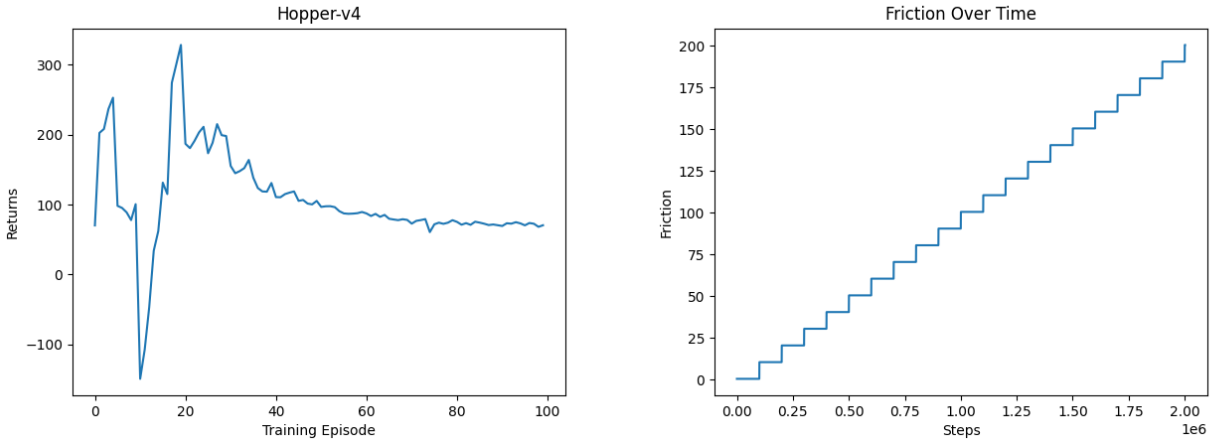

Figure 3. **Early environment change dynamics tests and their associated performance—stepwise increasing geometric friction:** Initial environment changing dynamics involved the continual modification of a particular parameter (in this particular case friction). Training episode and steps are different labeling units for the same process.

the other L2 init method with a smaller regularization loss weighting of $\lambda = 10^{-3}$ learns quickly, outperforming the trained from scratch model.

We observe vastly different results when adjusting the hopper's body mass as per a very similar environment change agenda. Here, we again observe the benefit of training models starting on the low mass epoch, as the PPO model trained from scratch on the second epoch was not able to learn to achieve higher returns on either the high mass epoch (2nd in brown), nor the inference epochs at the end (9th and 10th in pink). In addition, although the L2 init model outperformed other models on the low and high geometric friction epochs, all models lost out to the PPO with CReLU activation function model on the high friction epochs. Also, all models were able to significantly improve their returns throughout each training epoch. In addition, the hybridized L2 init with CReLU activation approach outperformed every other model on the low mass epochs, and also tied CReLU's performance on the first unseen parameter value period (9th epoch). CReLU beat the hybrid approach and all other models on the last, large mass value inference period, and was one of the few models that continued to learn quickly as the epoch went on. We believe that there's a vast difference in performance between the two sets of parameters because of their influence on the hopper. Geometric friction

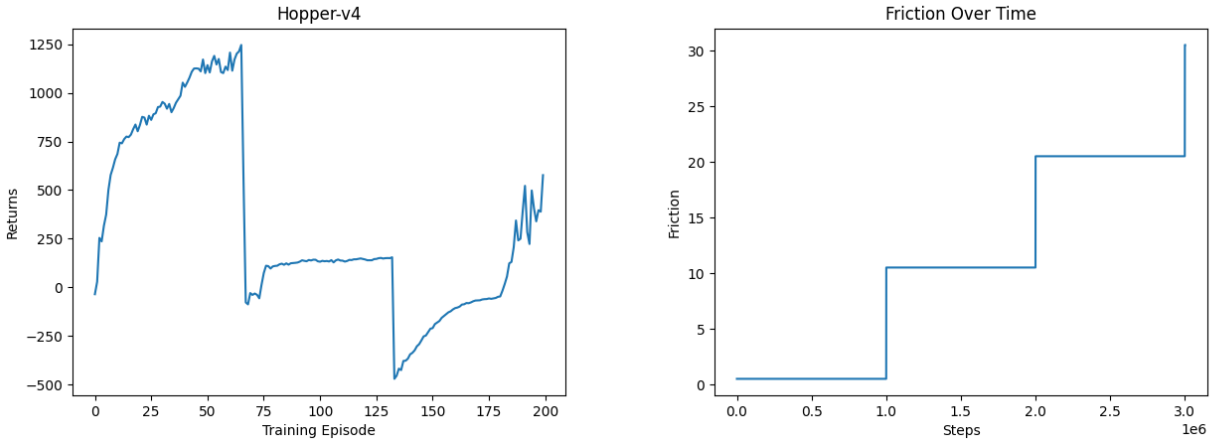

Figure 4. **Early environment change dynamics tests and their associated performance—large, delayed step increases:** Initial environment changing dynamics involved the continual modification of a particular parameter (in this particular case friction). Training episode and steps are different labeling units for the same process.

directly opposes the actions that the PPO agent makes in each state (outputting torques). However, the body mass has a slightly indirect effect, allowing for larger changes in mass while still resulting in the same outcomes, albeit to slightly different magnitudes, in the agent.

Throughout both of these models, a PPO model trained with a completely vanilla MLP as the actor and critic method, with the same size as all other models, underperformed most methods in the high friction and high mass value regimes. This showcases the classic architecture's lack of plasticity. As exemplified by the average returns from body mass, the model hyper optimizes for any particular environmental setting it is introduced. This not only erases the effects of previously learned features from previous environmental settings, but decreases the models overall ability to utilize its time within a particular environment for general learning (which it forgoes for particular environmental learning since policy optimization is 'environmentally'-greedy). This can be seen due to the lack of performance on the final parameter update (the out of distribution environment). However, when compared to other models, it explicitly shows the improvement that implementing continual learning methods into MLPs for continual reinforcement learning can have.

We now discuss how parameter change agendas with a final inference value (10th epoch) that is over 5 times the largest value previously seen in training can affect training, shown in Figure 6. First, we discuss geometric margin. The training process is much more smooth for almost all models, even throughout the beginning impulse train portion of the parameter agenda. Models largely continue learning in the second, high geometric margin epoch at the same pace as the first epoch. In this case, it's clear that geometric margin has less of a direct effect on the rewards given after the PPO agent's actions. Interestingly, though, we observe that the highest variance and lowest (across seeds) average return performance were attributed to the CReLU and L2 init with $\lambda = 10^{-3}$, two of the most performant methods for the smaller inference variance cases in Figure 5. Instead, we observe that L2 init with a larger $\lambda = 10^{-2}$ weight performs the best, followed by the hybrid CReLU and L2 init approach. This is consistent even across high geometric margin epochs. In the last two inference epochs, the models slowly learn, but collapse when faced with the large inference geometric margin value.

Lastly, we have the anti-gravitational component parameter change. We observe that the vanilla MLP PPO model outperforms all other continual learning technique models. This insinuates that our approaches may work well in environments that change drastically, but for static environments, PPO still beats out other variants of its architecture. Among the continual learning models, we observe that the L2 init with the smaller regularization weight and CReLU perform the best. We also see that vanilla PPO trained from scratch on the high antigravity (2nd) epoch, as well as the last two inference values, struggles to learn about the environment, another reinforcing observation to our heuristic that training on "easier" (less resistance to agent action) contact environments may help long term training phase returns.

5. Conclusion and Future Work

Traditional neural network-based value function and policy representations have failed to perform in reinforcement learning problems with changing environments. We imple-

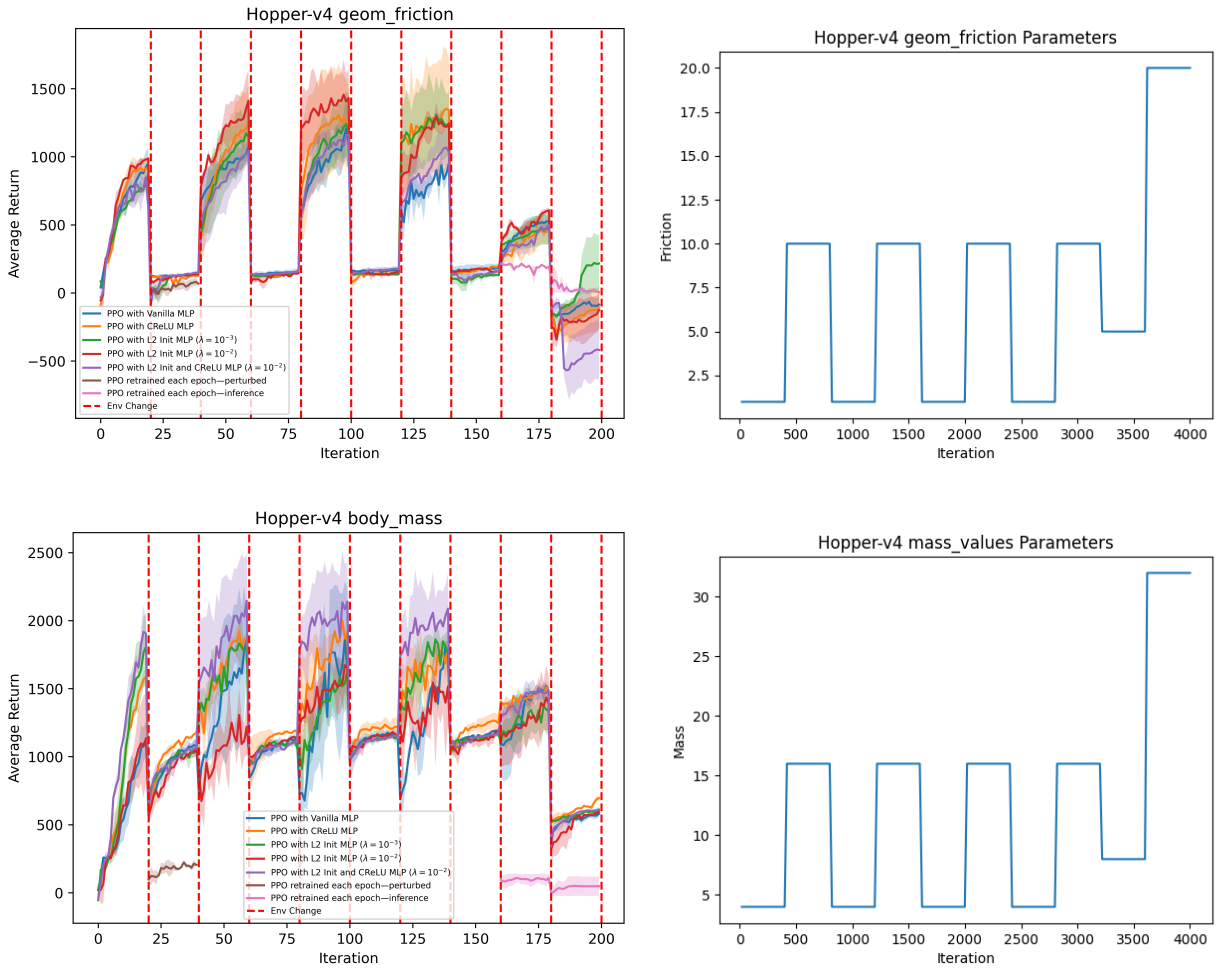

Figure 5. **Suite of environmental changes with associated environment change agendas and their respective influence on policy returns—small variation in inference cases:** Using an agenda of environment changes with respect to a particular environmental setting (shown in the right column), we tested the average returns derived over four initial seeds (shown in the left column) to assess the ability of PPO on Hopper-v4 to showcase plasticity. Vertical red lines on the left column indicate an environment shift that should reflect the agenda in the right column. Here, we varied geometric friction (at the joints) and the mass of the hopper by a small amount, testing an out of distribution value only around twice as large as the maximum value we perturbed the parameters to during the first 8 iterations.

mented a variety of approaches that utilize advancements in neural network plasticity to achieve improved continual learning capacity in the Hopper v4 environment. In particular, we employed a parameter change agenda that alternated between two values of each Hopper parameter being tested before switching to two values that had not been seen before, with one being in distribution and the other being far above the maximum parameter value observed yet. We observed that PPO models with CReLU activations and hybrid PPO models with CReLU and regenerative regularization ($\lambda = 10^3$) performed well in environments that were more drastically changing. For environments that changed less drastically, regenerative regularization ($\lambda = 10^{-3}$) and the aforementioned hybrid approach performed best.

In the future, we hope to implement continual backpropagation to remove importance on parameters that are not being used as the environment changes, a more drastic architectural change that may yield better learning results under large environmental parameter perturbations. We could also try to implement more smooth, exponential transitions between the beginning values in each parameter agenda.

References

Abbas, Z., Zhao, R., Modayil, J., White, A., and Machado, M. C. Loss of plasticity in continual deep reinforcement learning, 2023.

Abel, D., Barreto, A., Van Roy, B., Precup, D., van Has-

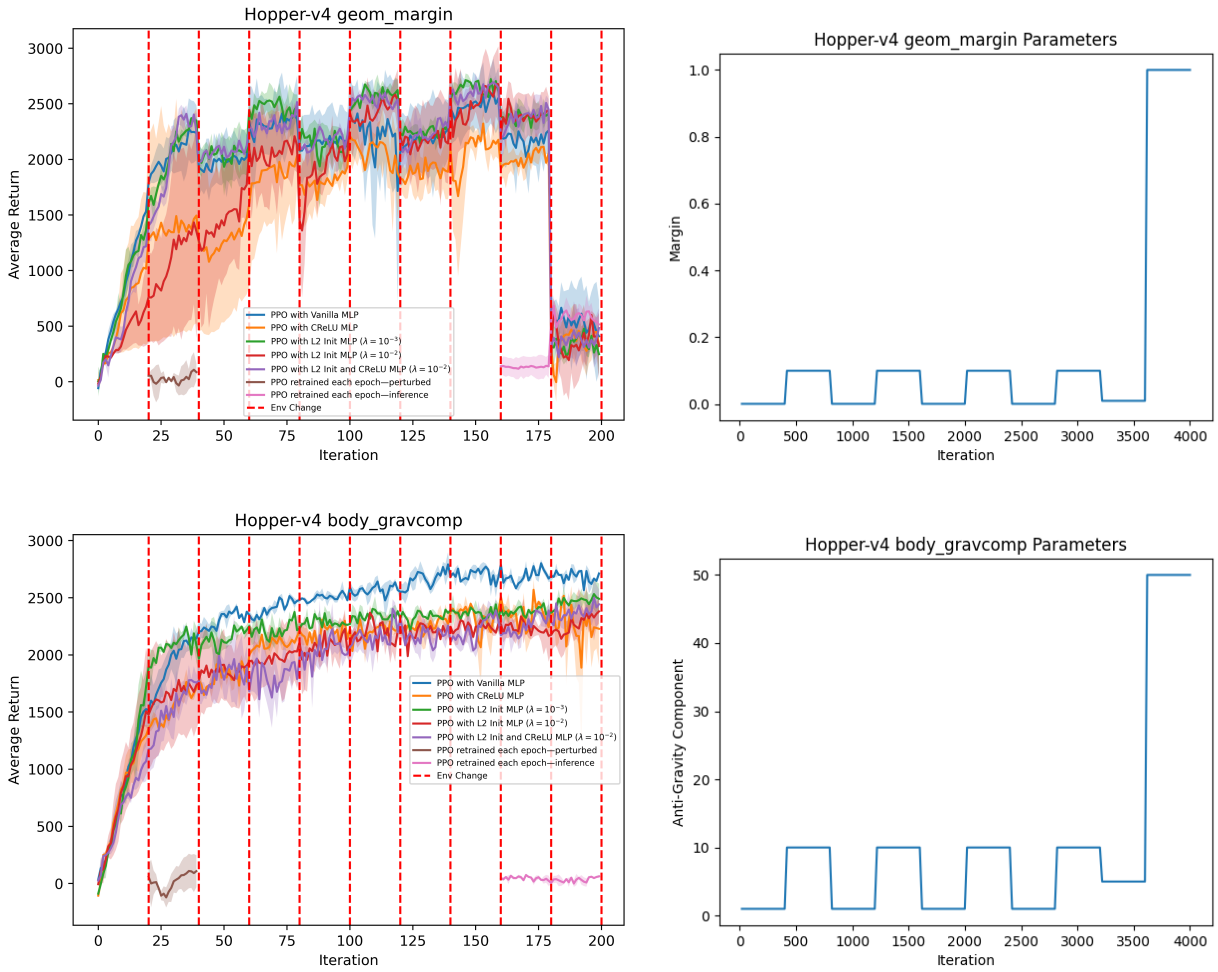

Figure 6. **Suite of environmental changes with associated environment change agendas and their respective influence on policy returns—large variation in inference cases:** Using an agenda of environment changes with respect to a particular environmental setting (shown in the right column), we tested the average returns derived over four initial seeds (shown in the left column) to assess the ability of PPO on Hopper-v4 to showcase plasticity. Vertical red lines on the left column indicate an environment shift that should reflect the agenda in the right column. Here, we varied geometric margin and an anti-gravity factor by a large amount, testing an out of distribution value between 5 and 10 times as large as the maximum value we perturbed the parameters to during the first 8 iterations.

selt, H. P., and Singh, S. A definition of continual reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Buesing, L., Weber, T., Racanière, S., Eslami, S. M. A., Rezende, D. J., Reichert, D. P., Viola, F., Besse, F., Gregor, K., Hassabis, D., and Wierstra, D. Learning and querying fast generative models for reinforcement learning. *CoRR*, abs/1802.03006, 2018. URL http://arxiv.org/abs/1802.03006.

Clavera, I., Nagabandi, A., Fearing, R. S., Abbeel, P., Levine, S., and Finn, C. Learning to adapt: Meta-learning for model-based control. *CoRR*, abs/1803.11347, 2018. URL http://arxiv.org/abs/1803.11347.

Dohare, S., Sutton, R. S., and Mahmood, A. R. Continual backprop: Stochastic gradient descent with persistent randomness, 2021.

Dulberg, Z., Dubey, R., Berwian, I. M., and Cohen, J. D. Having multiple selves helps learning agents explore and adapt in complex changing worlds. *Proc Natl Acad Sci U S A*, 120(28):e2221180120, July 2023.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400, 2017. URL http://arxiv.org/abs/1703.03400.

Kumar, S., Marklund, H., and Roy, B. V. Maintaining plas-

ticity in continual learning via regenerative regularization, 2023.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning, 2015.

Lyle, C., Zheng, Z., Nikishin, E., Avila Pires, B., Pascanu, R., and Dabney, W. Understanding plasticity in neural networks. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 23190–23211. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/lyle23b.html.

Padakandla, S. A survey of reinforcement learning algorithms for dynamically varying environments. *ACM Comput. Surv.*, 54(6), jul 2021. ISSN 0360-0300. doi: 10.1145/3459991. URL https://doi.org/10.1145/3459991.

Padakandla, S., K. J., P., and Bhatnagar, S. Reinforcement learning algorithm for non-stationary environments. *Applied Intelligence*, 50(11):3590–3606, Nov 2020. ISSN 1573-7497. doi: 10.1007/s10489-020-01758-5. URL https://doi.org/10.1007/s10489-020-01758-5.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Shang, W., Sohn, K., Almeida, D., and Lee, H. Understanding and improving convolutional neural networks via concatenated rectified linear units, 2016.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, Oct 2017. ISSN 1476-4687. doi: 10.1038/nature24270. URL https://doi.org/10.1038/nature24270.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.