Coaching with Controllers: A Novel Paradigm for Merging Classic Control with Learning

Zongqiang Pang, Liping Bai

***Abstract*—In this paper, we propose a controller based coaching scheme in order to accelerate reinforcement learning (RL). Increasingly, control community and RL community have realized that they are complimentary additions to one another. Existing control strategies would make the learning process more efficient while learning extends control to less scripted scenarios. We believe one productive way to merge classic control with learning is through the concept of coaching. World class atheletes forge their skillsets with a team of coaches, who provide conditions for the atheletes to collect critical experiences. This relationship can be constructed between the classical controllers and RL agents as well. While classical controllers might be inferior to RL agents in terms of control strategies, they can provide guidance for expedited data collection. We devise and implemente a method for "Controller Based Coaching", derived from our observations of the athletic training process. Experiments are conducted in Mujoco locomotion environments and we conclude from the data that when the coaching structure is set at its goldilocks spot, agents training can be accelerated, in some cases significantly, yielding uncompromised training results in the mean time. This is an important proof of concept that classical controller based coaching can be a novel and effective paradigm for merging control with learning and warrants further investigations along this direction. All the code and data can be found at: github.com/BaiLiping/ControllerBasedCoaching**

***Index Terms*—Reinforcement Learning, Control, L4DC**

I. Introduction

REINFORCEMENT Learning (RL) is a cyle between interaction with the environment based on current understandings and refinement of that understanding based on observations. RL agents methodically extracting information from experiences, gradually bounding system models, policy distributions or cost-to-go approximations to maximize the expected rewards along a sequence of actions. While the field of RL routinely generate jaw-dropping results [1][2][3][4][5], there are aspects of its methodology that are inherently inefficient and the focus of current researches is to reformulate RL into its more effective form.

There are three angles of attack in order to address the inefficiencies. First, RL researchers who are well versed in statistics and information theories approach the problem by devising ever more sophisticated statistical processes to push information extraction to its limit.[6][7][8][9][10][11][12][13][14][15][16][17] Second, control researchers who are capable of wielding control theories and optimization techniques provide a systematic perspective to RL which is a dearth of analytical frameworks.[18][19][20][21][22][23][24][25][26][27][28][29] Third, a immitation or teaching structure to incorporate the best control strateges into RL training process. [30][31][32][33][34]

In this paper, we propose a fourth angle of attack: classical controllers based coaching. Professional athletes don't get where they are via trial-and-error. Their skillsets are forged through a series of theoretically and empirically proven training techniques that seek to maximize training efficiency. Based on this observation, we construct a coaching relationship between classical controllers and RL agents.

The objective of a classical controller based coach is to facilitate data collection on critical states, much like a coach guides athletes towards essential experiences instead of repeating steps that merely pave the way. We ascribe the observed training acceleration brought about by our approach to this expedited data collection on critical states. We would elaborate on our ideas further in the subsequent sections.

We implemented the proposed active boundary scheme on various OpenAI Gym environments as a proof of concept and we concluded that when the boundaries are set appropriately, accelerated learning can be achieved and the resulting agents have indistinguishable evaluation scores compare to the agents trained with normal methods. We have confidence in our conclusion since while randomness might underline one set of data, it is highly unlikely that randomness alone can explain the same observations made in a different set of experiments. Our result indicated the feasibility of adopting human training techniques into RL setup and could be the beginning of other fruitful researches in this direction.

In section II we introduce reinforcement learning in the language of control. In section III we detail our proposed active boundary scheme for accelerated RL agent training. In section IV, we present the results of all our experiments. While the experiments are conducted in OpenAI Gym simulated environment, we consciously designed things in a way such that it can be conveniently implemented in the physical world as well. In the final section, we conclude what we have learned from our experiments and layout directions for further research.

II. Reinforcement Learning For Control Researchers

A. Note on Notation and Nomenclature

The first barrier stands between control researchers and RL is the jungle of notations. At this point, just about every researcher has his own symbolic system. As noted by Warren Powell in his 2014 introductory paper on Approximate Dynamic Programming: "Somewhat frustratingly, these communities have also developed different vocabularies and different notational

Nanjing Unversity of Posts and Telecommunications, College of Automation & College of Artificial Intelligence, Nanjing, Jiangsu,210000 China email:zqpang@njupt.edu.cn

systems." Yet in the same breath, Powell introduced a new set of customized notation. [35] Here, we try to point out the differences to make life easier for control researchers. Control communities use the notation system introduced by Lev Pontryagin. The state is denoted by $\mathcal{X}$, conventionally a letter representing the unknown. Action is denoted by $\mathcal{U}$, the first letter of Russian for action. The dynamics and stochasticity is captured by physical model constraints $x_{t+1} = f(x_t, u_t, e_t)$ where e denotes the noise of a system. The objective is usually to minimize the cost function $\mathcal{J}(.)$. Reinforcement Learning communities adapt their notation system from the field of Operations Research. State is denoted $\mathcal{S}$, Action is denoted $\mathcal{A}$. The system dynamics and stochasticity is captured via transition matrix $\mathcal{P}$ of a Markov Decision Process. The objective of RL is the maximize the reward function $\mathcal{R}(.)$.

Richard Sutton uses terminologies that can be overloaded and confusing, such as Monte Carlo, Temporal Difference(TD), in his book Introduction to Reinforcement Learning [36]. Although the control community also has to deal with the sequential aspect of a problem, the terminologies used are much more down-to-earth and straightforward, such as one-step look ahead, n-steps look ahead. In the context of RL, Monte Carlo simply means update with data collected from an entire episode. Temporal Difference means update with data collected with one step. TD(λ) means update with data collected with λ steps.

B. From Optimization To Learning

Reinforcement Learning is not a new subject, control researchers probably know it by the name Approximate Dynamic Programming (ADP) or Neuro Dynamic Programming (NDP). When the first batch of reasonable RL results was introduced, it was met with coldness by people in control. The preface of Bertseka's book Neuro Dynamic Programming provides a good sample of how RL was perceived by the control community back in 1996: "...These methods (Reinforcement Learning) were aiming to provide effective suboptimal solutions to complex problems of planning and sequential decision making under uncertainty, that for a long time were thought to be intractable. Our first impression was that the new methods were ambitious, overly optimistic, and lacked a firm foundation. Three years later, after a lot of study, analysis, and experimentation, we believe that our initial impressions were largely correct." [37]

Today, the theoretical foundation of RL is as bleak as it was in 1996, but its deficiencies can be easily papered over with the advant of Neural Networks and ever more impressive computational capacities. Before 2010, the predominant function approximators are handcrafted kernels, where feature spaces transform nonlinear functions into linear space. Today, the default function approximator is Deep Neural Network and its variants. In the computation front, Graphic Process Unit (GPU) based parallelization is accessible to anyone who is proficient with C/C++[38], and software packages such as PyTorch[39] and Tensorflow[40] made this task even more trivial.

In order to better introduce RL to control researchers, we built on Benjamin Recht's presentation[41], framed RL in the language of optimization. Reinforcement Learning is an umbrella term, which encompasses distinctive genres of learning.

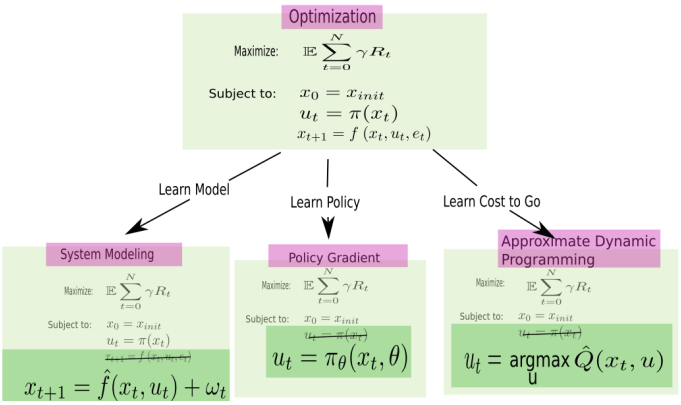


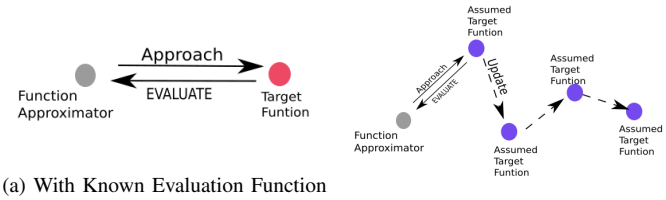
Fig. 1: From Optimization to Learning

Learning in the context of RL means updating parameters of the function approximators. Categorically speaking, there are three avenues where learning finds its way into optimization as shown in Figure1.

Model-Based or Model-Free learning refers to whether or not learning is used to approximate the system dynamics function. If there is an explicit action policy, it is called on-policy learning. Otherwise, the optimal action would be implicitly captured by the Q value function and that would be called off-policy learning instead. Importance sampling allows "limited off-policy learning" capacity, which allows data reuse in a trusted region. Online learning means interleaving data collection and iterative network parameters update. Offline learning means the data is collected in bulk first and then the network parameters are set with regression computation. Batch learning, as the name suggested, is in between online and offline learning. An agent would first generate data that fill its batch memory and then sample from the batch memories for iterative parameter update. New data would be generated with the updated parameters to replace older data in the memory. This taxonomy is somewhat outdated now. When Richard Sutton wrote his book, the algorisms he had in mind fall nicely into various categories. Today, however, the popular algorisms would combine more than one route to derive superior performance, and can't be pigeonholed.

We will not go into the details of various RL algorithms and the zoo or acronyms that come with it, as that is beyond the scope of this paper. PPO agents were chosen for all our experiments because the learning curves generated by PPO agents are smoother compare to other agents. The goal of our paper is to indicate the proposed active boundary method works, which is easier to be shown with smoother curves. However, we would like to point out two crucial concepts that undergird the feasibility of reinforcement learning.

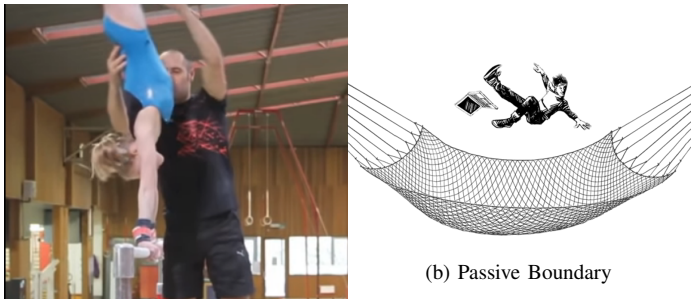
1) Bellman Update: Reasoning backward or Dynamic Programming forms the basis of reinforcement learning. If we know the desired final state and system dynamics then we can simply compute backward on the penultimate step. Set the penultimate step as the final step, then repeat the process until the action strategy for the initial state is obtained. This approach, Bellman Update, would work if two prerequisites are met: first, the system dynamics function is known; second,



(a) With Known Evaluation Function

(b) Bootstrap

Fig. 2: Convergence through Bootstrap



(a) Active Boundary

(b) Passive Boundary

Fig. 3: Difference Between Active Boundary and Passive Boundary

the dimensions of observation space and action space are small. Neither would be true for the problems we encounter in the real world.

When the dynamics are unknown yet the dimensions of the problem are manageable, the cost-to-go lookup table can be constructed with sampling methods. As mentioned earlier, the way that the lookup table is updated with the sample can be categorized by the horizon of the data collection. If the lookup table is updated with data collected from an entire episode, it is called the Monto Carlo method. If the lookup table is updated every step, it is called Temporal Difference method, and there are various methods in between these two extremes. When the dimensions of the problem are large in addition to unknown dynamics, function approximators are used for the system model, the policy function, or the cost-to-go function.

Immediately, control researchers can spot the deficiency with the latter case. For backward reasoning to work properly, the notion of optimality has to be satisfied either through known dynamics function or robust sampling methods, yet when the dynamics are unknown and the dimensions large, the notion of optimality can no longer be guaranteed through sampling. This is where the expertise of statisticians and information theoriests comes in and the research topic of exploration-exploitation trade-off.

2) Bootstrap: Another fundamental concept for RL is convergence through bootstrap. Instead of asymptotically approaching a known target function, bootstrap methods approach an assumed target first and then update the target assumption based on collected data. This process is illustrated by Figure2. Intuitively, when estimation functions are evaluated with assumption instead of the true value, things could just run around in circles and never converge. That is indeed the case before the introduction of DQN[42]. As a matter of fact. the stability and convergence achieved in that paper are more of a reflection on the techniques and know-how of the DeepMind team. Initially, those results can not be easily replicated by other labs. Even today, instability in RL is still a major research focus.

III. Expedited Data Collection on the Critical States with Active Boundary

As we pointed out in the introduction, human has amassed vast amounts of empirically proven techniques for accelerated skill acquisition. In this paper, we focus on the idea of expedited data collection on critical states with active boundaries, where critical states refer to the ones that are essential for skill acquisition.

Focused training with critical states is the bread and butter for athletes. Chess players start their training with endgames because that allows them to dwell on critical steps which illustrate the reasoning behind chess tactics, instead of wasting time on the "mechanical/maintenance moves" that lead up to a strategic position; Tennis coaches feed athlete balls with a specific angle and spin speed to force he with a backhand or forehand response. Examples like this are abundant in the world of sports. Active boundary is one way to implement focused training with critical states.

When the athlete is about to deviate from optimal action, an attentive coach would apply force to course-correct on behalf of the athlete. This way, the athlete would have experienced the right way to handle that state as opposed to just falling. As indicated by Figure 3, a coach can implement an active boundary because he can apply the appropriate force in due time. A safety net, on the other hand, is a passive boundary, since it does nothing other than catch people during a fall and no information on the critical state is generated in that process. The distinction between active and passive boundary is important because while prevention of premature termination is an implicit outcome of active boundary, it is by no means its primary goal. The protective property of active boundary stems from the fact that when a critical state is handled poorly, a terminal state would ensue. Therefore, the protective property is a by-product of directing the agents towards critical states. We would further accentuate the distinctions in the following section where we provide details of our implementation of active boundary.

IV. Implement Active Boundary in OpenAI Gym

We implemented our active boundary experiments in OpenAI Gym[43] as a proof of concept. We are cognizant of the fact that eventually, things should apply to a real-world scenario. Therefore, we designed the active boundary in a manner such that it can be swiftly adapted to physical implementation. Agents are implemented with tensorforce[44]. We sorted out the observation space and action space for each environment and recorded them in the code of each experiment.

As we suggested in the previous section when agents reach the critical state, "correct actions" would be applied by the active boundary to the agent, hence the name "active". Much

like a coach would guide athletes through the most crucial step of the training. We also designed a penalty term associated with active boundary to discourage its use. If the agent act flawlessly at the critical state, no penalty would be applied. On the other hand, if the agent requires active boundary intervention, a penalty term would be registered.

The positions and penalty terms are set in an ad hoc fashion in our experiments. At this point, we do not have an analytical frame that can guide us in making such decisions. We choose three positions for the boundary during each experiment and 4-6 penalty terms ranging from 0 upwards. The rule of thumb we used for the penalty is that the punishment score should be comparable to the reward.

Our experiments are conducted over the following environments: CartPole, with discrete action space; Inverted Pendulum, with continuous action space; Inverted Double Pendulum; Hopper; Walker2D. We did not implement our approach to more sophisticated environments such as HalfCheetah, Humanoid, and HumanroidStandup due to hardware constraints. Yet we feel that for the data we collected, our approach has shown conclusive results. The following are our experiment setup and results.

The dashed black lines are the training results of agents with normal training, which functions as the control group and the benchmark. The colored lines are the results of agents trained with varying penalty terms. The average evaluation scores of the agents are presented at the upper left corner of the graph.

A. CartPole and Inveretd Pendulum

CartPole's Observation Space is the vector: [Cart Position, Cart Velocity, Pole Angle, Pole Angular Velocity]. The discrete action space for cartpole is: [Push Cart to the left, Push cart to the right]. The continuous action space for Inverted Pendulum is an action ranging from -1 to 1, where -1 means the actuator moves the cart to the left with maximum power and 1 means the actuator moves the cart to the right with maximum power. The maximum episode step number for cartpole and inverted pendulum are 500 steps and 300 steps respectively. The terminal state for cartpole is angle of absolute value greater than 24 degrees. The terminal state for the inverted pendulum is an angle of absolute value greater than 0.2 radius. The reward for cartpole and inverted pendulum are both 1 for each step and 0 for the terminal step.

The active boundary is implemented as a ring anchored on the cart, at the height of the middle of the pole. Touch sensors would be installed in order to detect contact between the pole and the ring. We propose three sets of rings each allows a maximum angle of 0.5, 0.7, and 0.9 of the terminal state angle in the cartpole environment and 0.1, 0.4, 0.5 radii in the inverted pendulum environment, as shown by Figure 4.

When contact between the pole and the active boundary is detected by the sensor, the cart would be pushed towards the appropriate direction depending on the sign of the angle. This step can be easily programmed by a microcontroller in physical implementation, and for implementation in simulation, it is just a matter of specifying a premeditated action. For detailed code, please refer to the code depository. Note that should the boundary is set without touch sensors and the corresponding actions, it would just be a passive boundary instead of an active one.

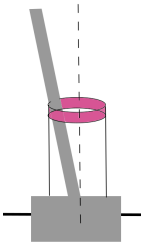


Fig. 4: CartPole/Inverted Pendulum active Boundary

A penalty term, which is chosen from vector [0,-5,-7,-10,-12,-15], would incur every time the pole contacts the active boundary. Our experiment result on cartpole are shown by Figure 5. The experiment result on inverted pendulum is shown by figure 6. As the data suggest, when the boundary is set too restrictively, it severely hinders the training. The best results is seen when the boundary is set at 10.8 degrees. After the agents are done training, we evaluate their performance in normal environments, the average evaluation scores are presented at the upper left part of the graphs. Our data suggest that during the evaluation, agents with accelerated training perform comparably to the agents with normal training.

B. Inverted Double Pendulum

The inveretd double pendulum has observation space of the following: [x position of the cart, $\sin(\theta_1)$, $\sin(\theta_2)$,$\cos(\theta_1)$,$\cos(\theta_2)$,velocity of x, angular velocity of θ_1, angular velocity of θ_2, constraint force on x, constraint force on θ_1, constraint force on θ_2]. θ_1 and θ_2 are the angles of the upper and lower pole perspectively.

The action space for Inverted Double Pendulum is an action ranging from -1 to 1, where -1 means the actuator moves the cart to the left with maximum power and 1 means the actuator moves the cart to the right with maximum power.

The Terminal state of double inveretd pendulum is when the y postion of the upper pole, which can not be observed by the agent, falls below 1.

We design the active boundary to be located at the same height as does the joint between the upper and the lower pole, as indicated by Figure 7. The boundary allows the joint to flex for 0.4, 0.5 and 0.6 radii respectively, each with the penalty parameters of a choice between 0,-3,-4,-5,-6,-7. When the joint extends to touch the active boundary, an action of -1 or 1 would be automatically applied to rectify the situation, depending on the sigh of the angles.

Our experiment result is shown by Figure 8. The conclusion is similar to that of cartpole and inverted pendulum results. When the active boundary is set at 0.4 radii, it hinders learning rather than facilitate it. When the boundary is set at 0.5 radii with a penalty parameter of -7, we see the most drastic acceleration of agent training. However, when the boundary is set at 0.6 radii, it seems to have given the agent too much flexibility since the result suggests that the acceleration is not as profound compare to that of the boundary set at 0.5.

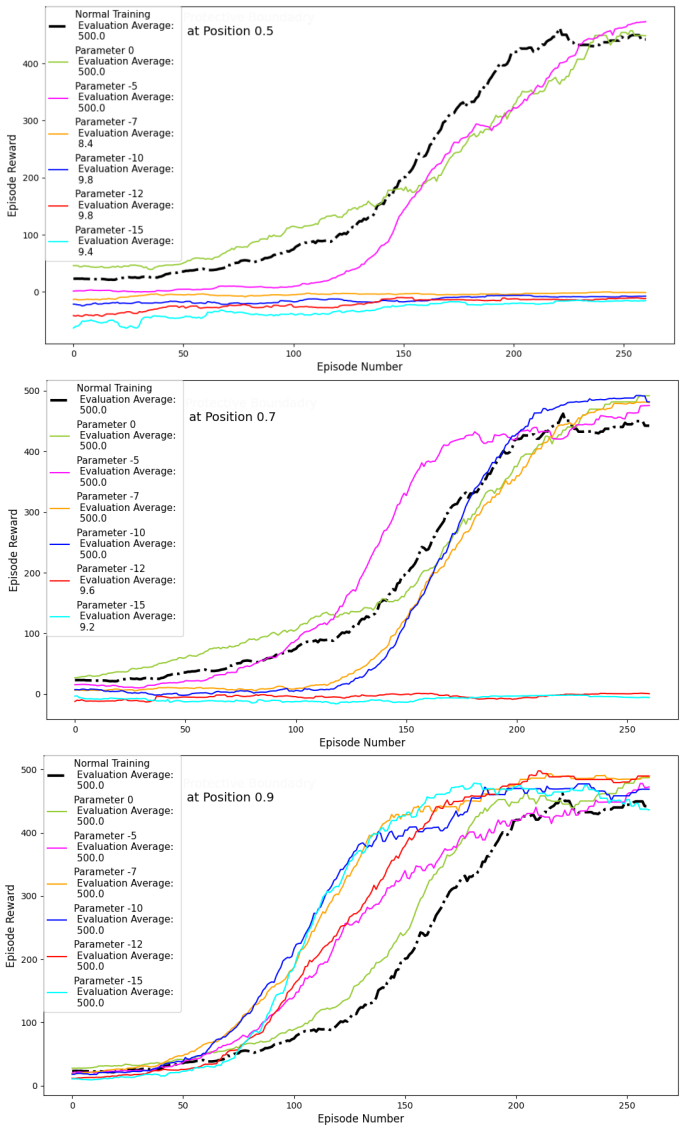


Fig. 5: CartPole Experiments

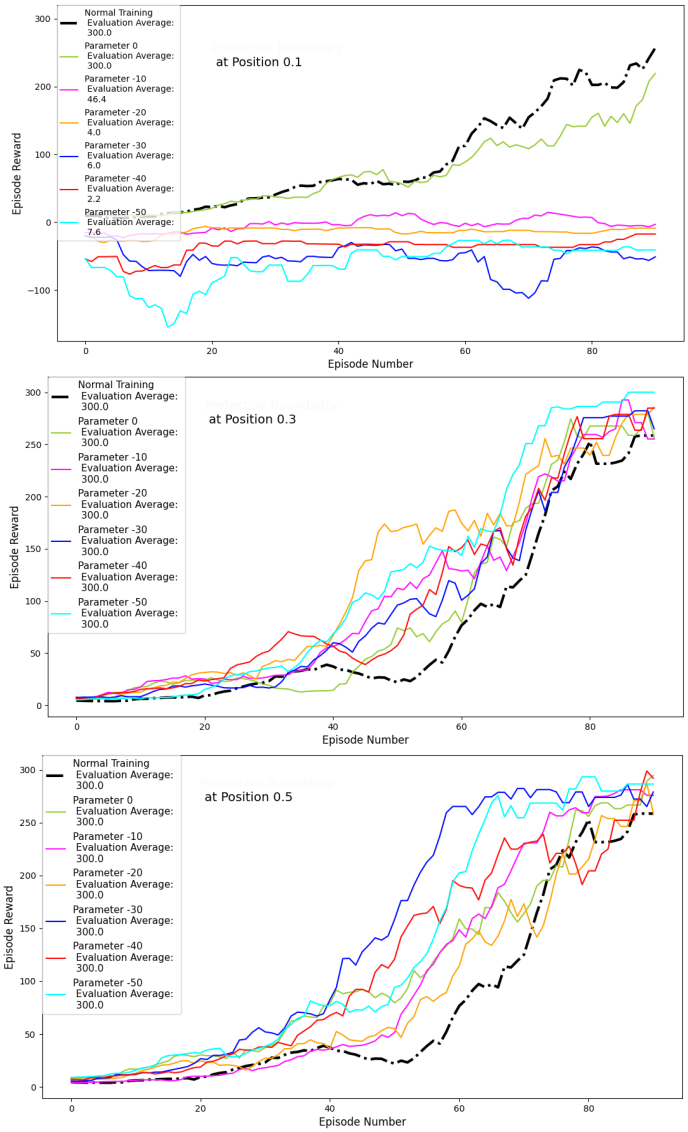


Fig. 6: Inverted Pendulum Experiments

C. *Hopper*

The observation space of hopper is the following vector: [z position, y position, thigh joint angle, leg joint angle, foot joint angle, velocity at x-axis, velocity at z-axis, velocity at y-axis, angular velocity of thigh joint, angular velocity of leg joint, angular velocity of foot joint].

The action space of hopper are three continuous action choices for three actuators [thigh actuator, leg actuator, foot actuator]. The range of for actuators are -1, which means apply force towards the negative direction with maximum power, and 1 which means apply force towards the positive direction with maximum power.

The terminal states for hopper are when the absolute y position is greater than 0.2.

There are more than one ways to implement active boundary in hopper environment. Because we want things to be applicable in real-world scenarios, we chose the implementation with the easiest setup. In our experiment, the active boundary is defined with the y-position at 0.05, 0.1 and 0.15, each with

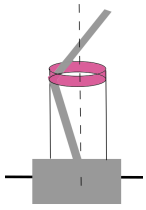


Fig. 7: Inveretd Double Pendulum active Boundary

the penalty parameters that is chosen from 0,-3,-5,-7,-10 as shown by Figure 9 When the agent touches the active boundary, we gently guide it slightly towards the center line, and the velocities terms are all set to zero. We seek to mimic how a coach would guide an athlete towards the desired trajectory during training. When the athlete is about to deviate off-course, the coach would take over and apply the appropriate force to slightly reset the trajectory. This can be implemented with an automatic harness in a physical experiment.

Our experiment result is shown by Figure 10. Similar to the

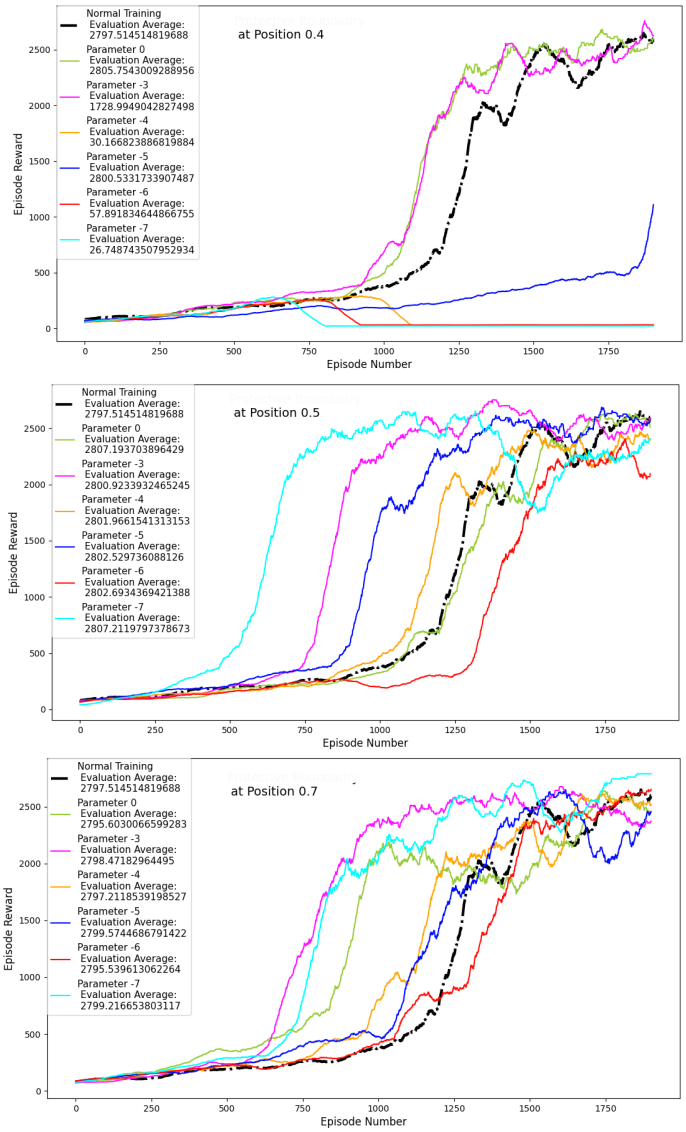


Fig. 8: Inverted Double Pendulum Experiments

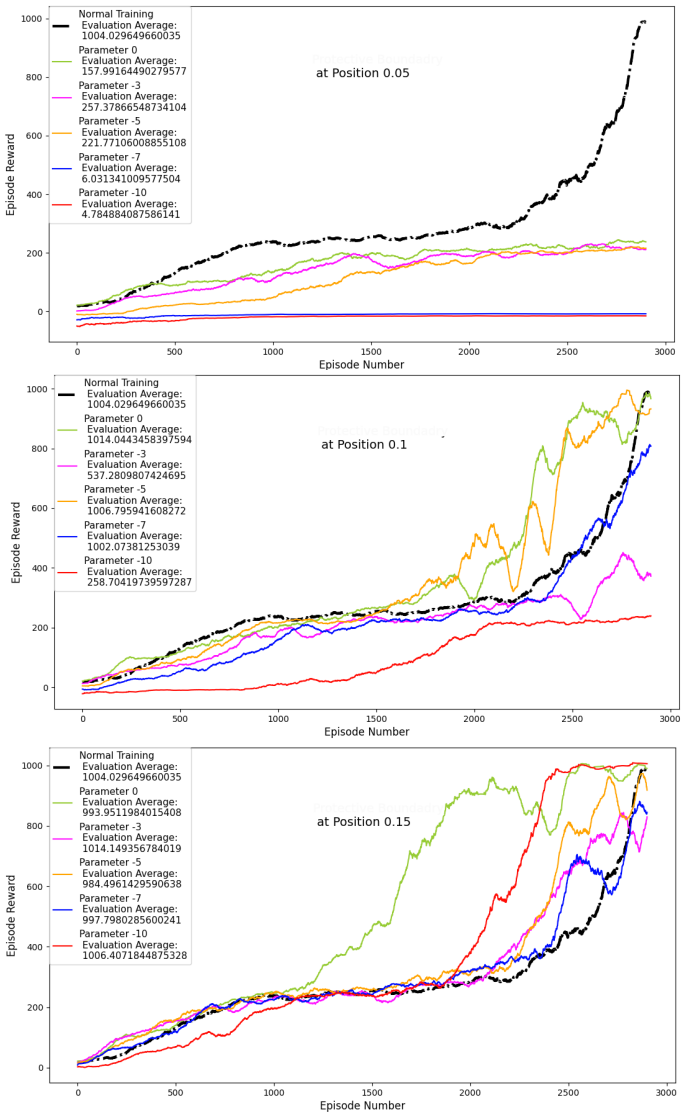


Fig. 10: Hopper

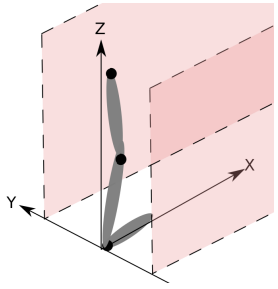


Fig. 9: Hopper active Boundary

conclusion drawn with previous data, when the active boundary is set too narrowly at 0.05 y position, it restricts the agent and almost completely halts the agent training. The best training acceleration is observed when the boundary is set at 0.15 y position.

D. Walker2d

The walker environment's observation space is the following vector: [z position, y position, right thigh joint angle, right leg joint angle, right foot joint angle, left thigh joint angle, left leg joint angle, left foot joint angle, velocity along x axis, velocity along z axis, velocity along y axix, angular velocity for right thigh joint, angular velocity for right leg joint, angular velocity for right foot joint, angular velocity for left thigh joint, angular velocity for left leg joint, angular velocity for left foot joint]

There are 6 actuators for the right thigh joint, right leg joint, right foot joint and left thigh joint, left leg joint, left foot joint perspectively. The range for the motors is -1, which means apply force towards the negative direction with maximum power and 1, which means apply force towards the positive direction with maximum power.

The terminal states are when the z position falls below 0.8 or rise above 2 or the absolute value of y position is greater than 1.

The active boundary is set with the y position at 0.3, 0.7

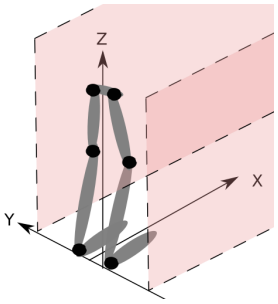


Fig. 11: Walker2d active Boundary

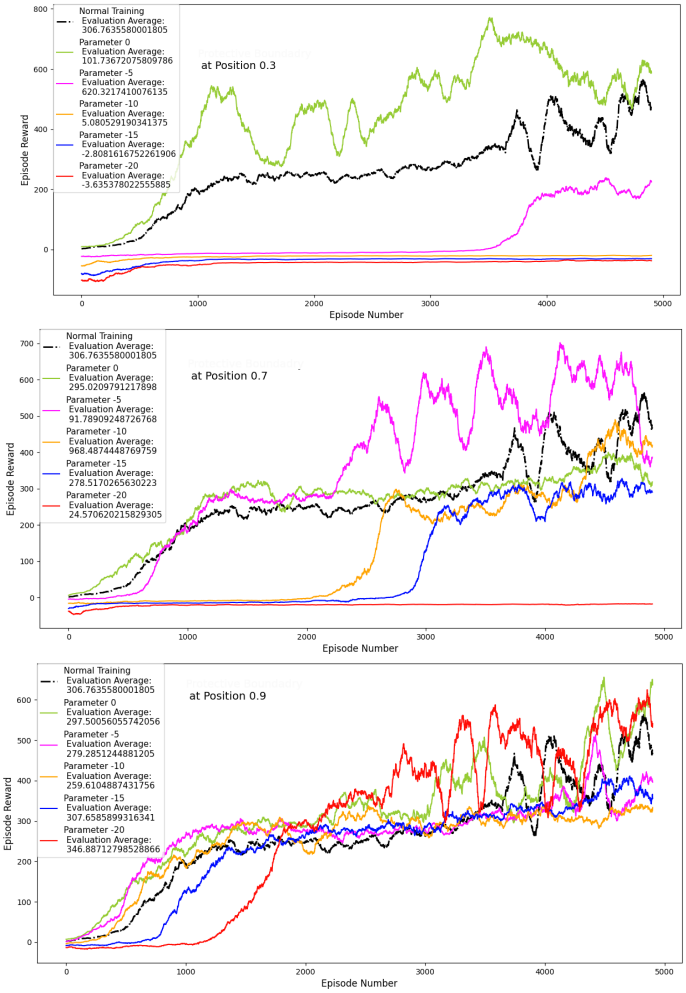


Fig. 12: Walker2D

and 0.9, each with the penalty parameters that is chosen from 0,-5,-10,-15,-20 as shown by Figure 11. Similar to the active boundary for hopper, when the agent touches the active boundary in the walker environment, we gently guide it slightly towards the central line, and the velocities terms are all set to zero in the meantime.

Our experiment result is shown in Figure 12. Yet again the experiment result suggests the same conclusion: when the boundary is positioned at 0.1, it acts as a barrier to training. When it is set at 0.7 positions, it could accelerate training with the appropriate penalty term.

V. Conclution

In this paper, we introduced the active boundary idea based on our observation of athletic training. Our proof of concept data suggests when the position and penalty associated with active boundary are set appropriately, accelerated learning can be achieved. Future research should focus on devising analytical tools that can systematically derive the best position and penalty for an active boundary.

Next step, we plan on extending the active boundary idea to the deep drone acrobat project [45]. The training of drone acrobat is done in simulation and then transferred to a real-world drone controller. We think our active boundary method can greatly accelerate the training of drone acrobats.

We believe our research opens door to a rich reservoir of potential researches along the direction of adapting proven human training techniques to RL environments. Human achieves superhuman feats not because of talent, but because of the meticulously engineered coaching tactics. Current researches in RL focus solely on the "athlete" side of the equation, i.e. how to build an efficient RL agent. But we feel "coach" is as important, if not more so. For instance, the endgames training strategy was deployed to educate chess players. During the training phase, the agent should start the game middle way to amass experience with critical states, instead of wasting time on steps leading up to the critical position. Coaches also challenge athletes, pushing them to experience difficult situations. "Coach" could even be implemented by independent neural networks, in addition to the agent, and trained to "teach" better. All these should be investigated by future researches.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *ArXiv*, vol. abs/1312.5602, 2013.

[2] M. J. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," in *AAAI Fall Symposia*, 2015.

[3] O. M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. W. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, pp. 20 – 3, 2020.

[4] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," *ArXiv*, vol. abs/1806.10293, 2018.

[5] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, 2020.

[6] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *NIPS*, 2016.

[7] C. Finn, I. J. Goodfellow, and S. Levine, "Unsupervised learning for physical interaction through video prediction," *ArXiv*, vol. abs/1605.07157, 2016.

[8] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 488–489, 2017.

[9] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros, "Large-scale study of curiosity-driven learning," *ArXiv*, vol. abs/1808.04355, 2019.

[10] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," *ArXiv*, vol. abs/1703.03400, 2017.

[11] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," in *ICLR*, 2018.

[12] O. Nachum, S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," *ArXiv*, vol. abs/1805.08296, 2018.

[13] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, "Feudal networks for hierarchical reinforcement learning," *ArXiv*, vol. abs/1703.01161, 2017.

[14] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," *ArXiv*, vol. abs/1505.05424, 2015.

[15] Y. Gal, J. Hron, and A. Kendall, "Concrete dropout," in *NIPS*, 2017.

[16] I. Dasgupta, J. X. Wang, S. Chiappa, J. Mitrovic, P. A. Ortega, D. Raposo, E. Hughes, P. Battaglia, M. Botvinick, and Z. Kurth-Nelson, "Causal reasoning from meta-reinforcement learning," *ArXiv*, vol. abs/1901.08162, 2019.

[17] J. Zhang, "Designing optimal dynamic treatment regimes: A causal reinforcement learning approach," in *ICML 2020*, 2020.

[18] E. Weinan, "A proposal on machine learning via dynamical systems," 2017.

[19] E. Dupont, A. Doucet, and Y. Teh, "Augmented neural odes," in *NeurIPS*, 2019.

[20] M. Betancourt, M. I. Jordan, and A. Wilson, "On symplectic optimization," *arXiv: Computation*, 2018.

[21] O. Nachum and B. Dai, "Reinforcement learning via fenchel-rockafellar duality," *ArXiv*, vol. abs/2001.01866, 2020.

[22] F. Luo, P. Li, J. Zhou, P. Yang, B. Chang, Z. Sui, and X. Sun, "A dual reinforcement learning framework for unsupervised text style transfer," in *IJCAI*, 2019.

[23] K. Wu and D. Xiu, "Data-driven deep learning of partial differential equations in modal space," *ArXiv*, vol. abs/1910.06948, 2020.

[24] G. Shi, X. Shi, M. O'Connell, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung, "Neural lander: Stable drone landing control using learned dynamics," *2019 International Conference on Robotics and Automation (ICRA)*, pp. 9784–9790, 2019.

[25] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," 2020.

[26] A. Mohan, N. Lubbers, D. Livescu, and M. Chertkov, "Embedding hard physical constraints in neural network coarse-graining of 3d turbulence." *arXiv: Computational Physics*, 2020.

[27] B. Lusch, J. N. Kutz, and S. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," *Nature Communications*, vol. 9, 2018.

[28] S. Bai, J. Z. Kolter, and V. Koltun, "Deep equilibrium models," *ArXiv*, vol. abs/1909.01377, 2019.

[29] F. de Avila Belbute-Peres, T. D. Economon, and J. Z. Kolter, "Combining differentiable pde solvers and graph neural networks for fluid flow prediction," *ArXiv*, vol. abs/2007.04439, 2020.

[30] W. B. Knox and P. Stone, "Interactively shaping agents via human reinforcement: the tamer framework," in *K-CAP '09*, 2009.

[31] ——, "Combining manual feedback with subsequent mdp reward signals for reinforcement learning," in *AAMAS*, 2010.

[32] X. Peng, P. Abbeel, S. Levine, and M. V. D. Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Trans. Graph.*, vol. 37, pp. 143:1–143:14, 2018.

[33] X. Peng, E. Coumans, T. Zhang, T. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," *ArXiv*, vol. abs/2004.00784, 2020.

[34] T. Paine, S. G. Colmenarejo, Z. Wang, S. Reed, Y. Aytar, T. Pfaff, M. W. Hoffman, G. Barth-Maron, S. Cabi, D. Budden, and N. D. Freitas, "One-shot high-fidelity imitation: Training large-scale deep nets with rl," *ArXiv*, vol. abs/1810.05017, 2018.

[35] W. Powell, "What you should know about approximate dynamic programming," *Naval Research Logistics*, vol. 56, pp. 239–249, 2009.

[36] R. Sutton and A. Barto, "Introduction to reinforcement learning," 1998.

[37] D. Bertsekas and J. Tsitsiklis, "Neuro-dynamic programming," in *Encyclopedia of Machine Learning*, 1996.

[38] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable parallel programming with cuda," *2008 IEEE Hot Chips 20 Symposium (HCS)*, pp. 1–2, 2008.

[39] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. Devito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[40] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zhang, "Tensorflow: A system for large-scale machine learning," *ArXiv*, vol. abs/1605.08695, 2016.

[41] B. Recht, "A tour of reinforcement learning: The view from continuous control," *ArXiv*, vol. abs/1806.09460, 2018.

[42] I. Osband, C. Blundell, A. Pritzel, and B. V. Roy, "Deep exploration via bootstrapped dqn," *ArXiv*, vol. abs/1602.04621, 2016.

[43] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *ArXiv*, vol. abs/1606.01540, 2016.

[44] A. Kuhnle, M. Schaarschmidt, and K. Fricke, "Tensorforce: a tensorflow library for applied reinforcement learning," Web page, 2017. [Online]. Available: https://github.com/tensorforce/tensorforce

[45] E. Kaufmann, A. Loquercio, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Deep drone acrobatics," *ArXiv*, vol. abs/2006.05768, 2020.