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

Zongqiang Pang, Liping Bai

***Abstract*—Reinforcement Learning (RL) is versitile yet the training process cumbersome while engineered controllers are relatively easy to compute but only applicable in scripted scenario. In this paper, we propose a controller based coaching scheme for RL in order to take the best from both worlds. Previous approaches for fusing classical control and RL are too restrictive since the quality of controllers end up inadvertantly constraints what is attainable by RL agents. We ask the question: is it possible to accelerate RL with even a badly designed controller? We draw inspiration from the relationship between world class atheletes and their coaches. In our approach, classical controller functions merely as a coach, whose job is to provide conditions for the RL agents to collect critical experiences, as opposed to a teacher, whose job is to provide the right answers. Experiments are conducted in Mujoco locomotion environments and we conclude from the data that when the coaching structure between classical controller and RL agents 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][30] Third, a immitation or teaching structure to incorporate the best control strateges into RL training process. [31][32][33][34][35]

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.

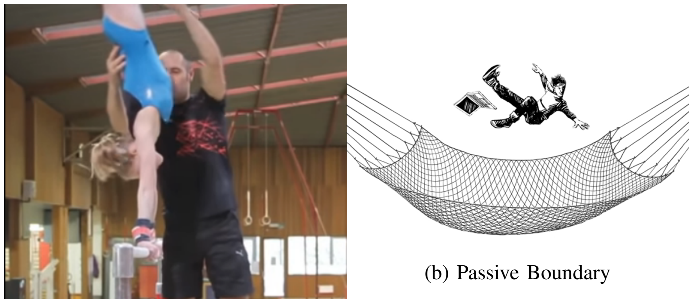
We are not the first group to realize that classical controllers can be used to guide RL agents, yet previous works are either implicit imitation based on the controllers[36] or RL is utilized to tune the controller parameters[37]. Our criticism to previous research is that if we want the RL agent to be more versitile than classical controllers, wouldn't the these approaches be inheriently restrictive? Much like we don't want world class atheletes to imitate his coaches, who is inferior to the atheletes in terms of performance. Our approach is much less restrictive and there is no danger that the controller would be the upper limit of the 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 our ideas further in the subsequent sections.

We implemente the proposed controller based coaching 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.

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


(a) Active Boundary

(b) Passive Boundary

Fig. 1: Difference Between Active Boundary and Passive Boundary

II. 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 1, 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.

III. Implement active Boundary in OpenAI Gym

We implemented our active boundary experiments in OpenAI Gym[38] 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[39]. 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 2.

When contact between the pole and the active boundary is detected by the sensor, the cart would be pushed towards the

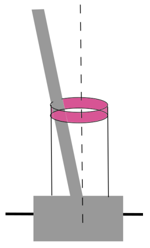


Fig. 2: CartPole/Inverted Pendulum active Boundary

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.

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 3. The experiment result on inverted pendulum is shown by figure 4. 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(θ_1), sin(θ_2),cos(θ_1),cos(θ_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 5. 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 6. 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

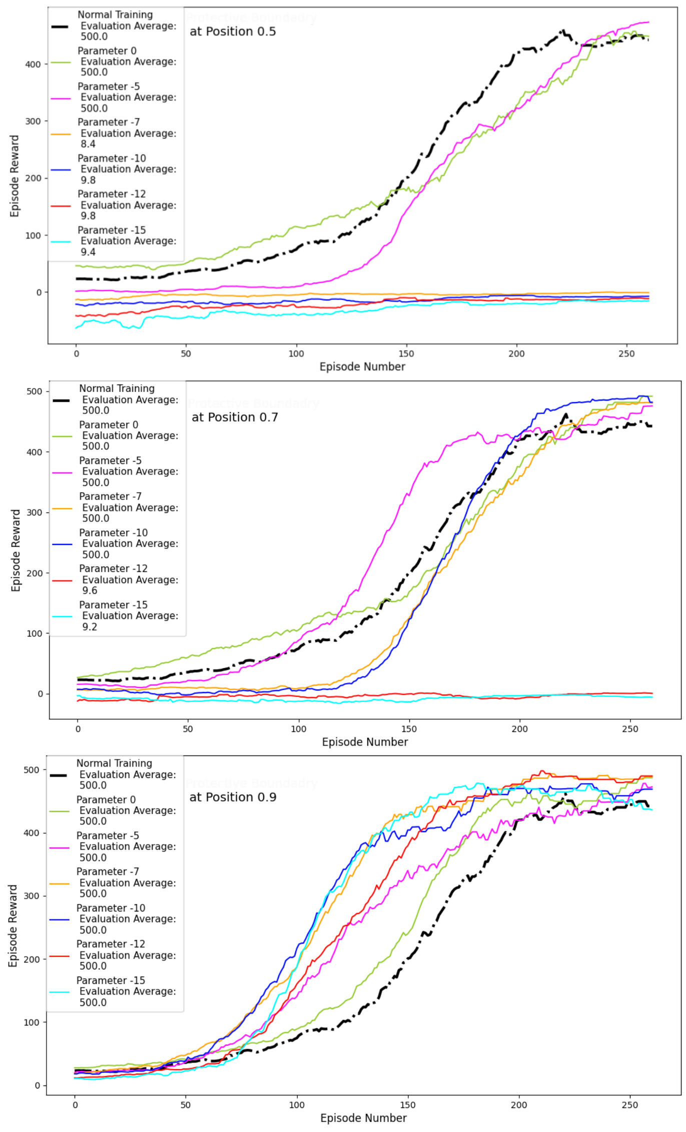


Fig. 3: CartPole Experiments

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.

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.

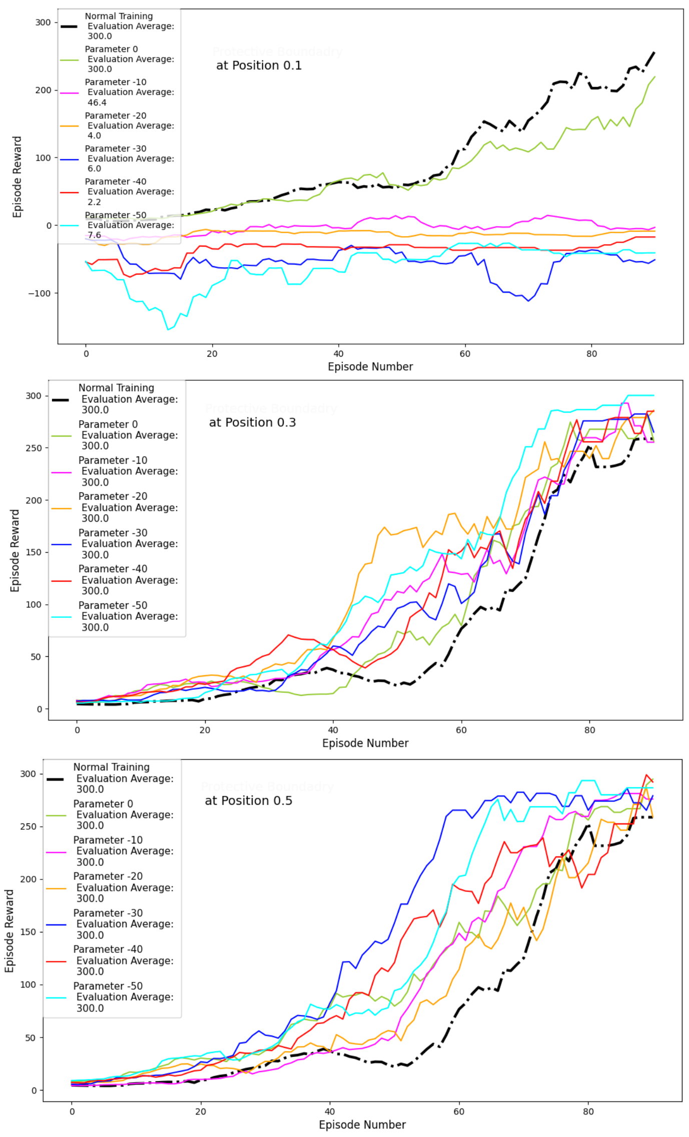


Fig. 4: Inverted Pendulum Experiments

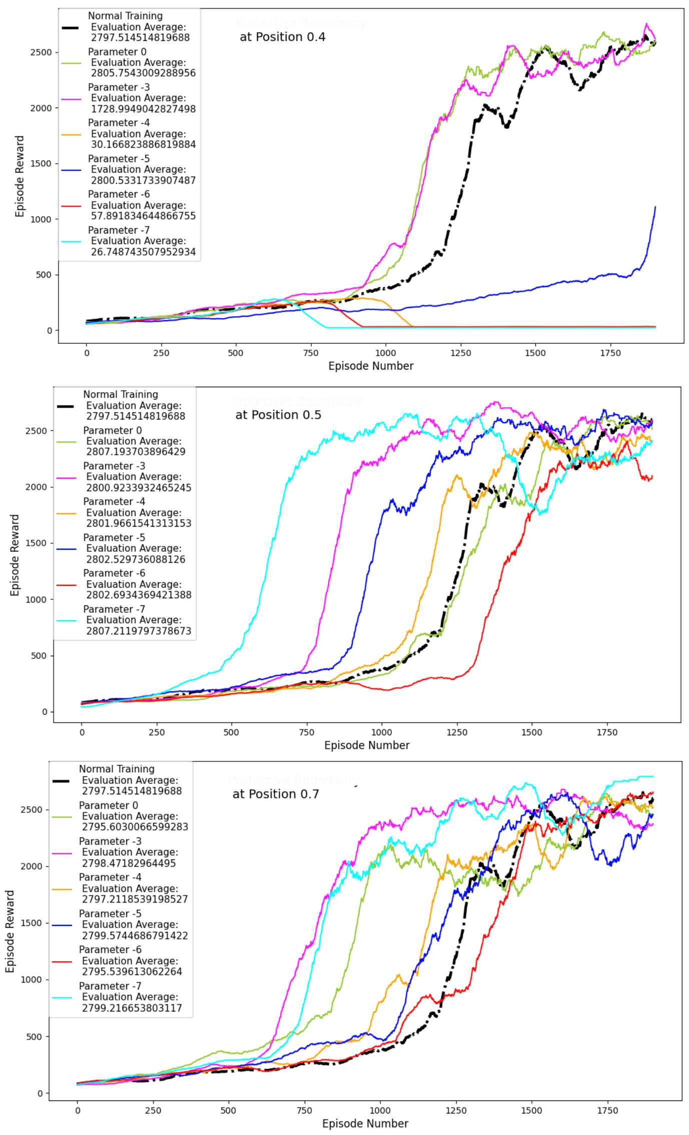


Fig. 6: Inverted Double Pendulum Experiments

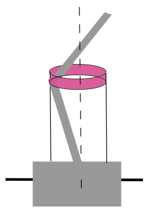


Fig. 5: Inveretd Double Pendulum active Boundary

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 the penalty parameters that is chosen from 0,-3,-5,-7,-10 as shown by Figure 7 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 8. Similar to the 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

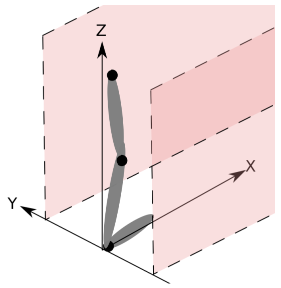


Fig. 7: Hopper active Boundary

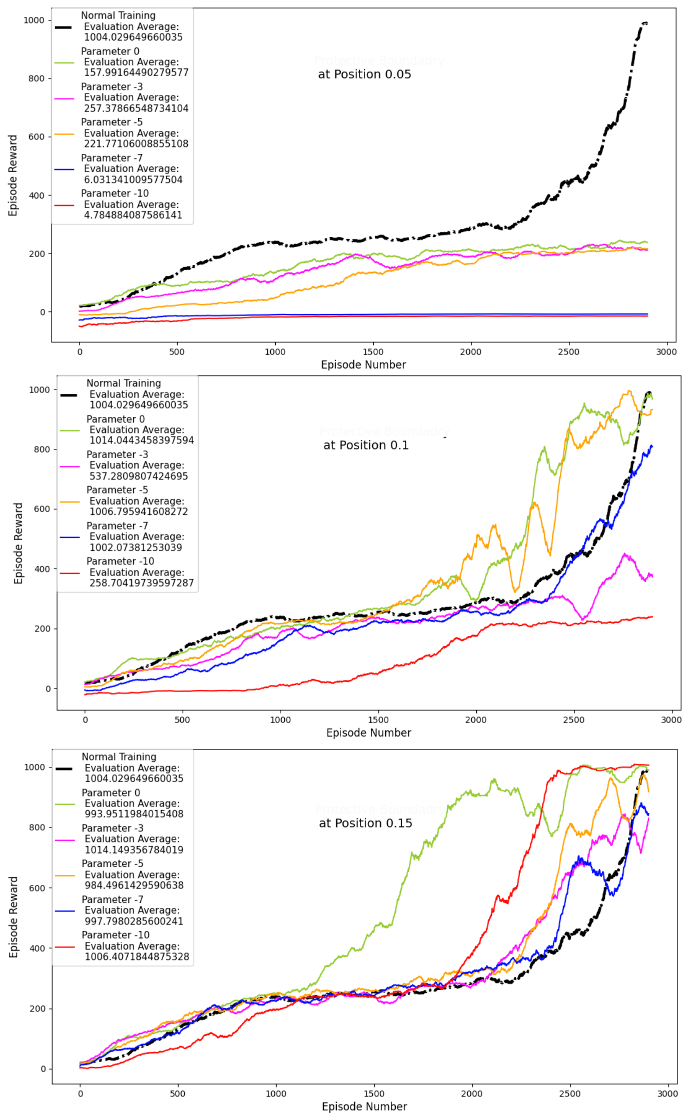


Fig. 8: Hopper

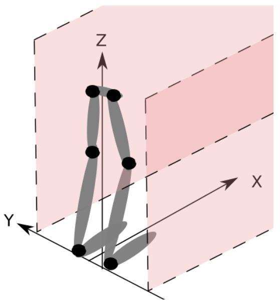


Fig. 9: Walker2d active Boundary

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 and 0.9, each with the penalty parameters that is chosen from 0,-5,-10,-15,-20 as shown by Figure 9. 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 10. 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.

IV. 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 [40]. 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

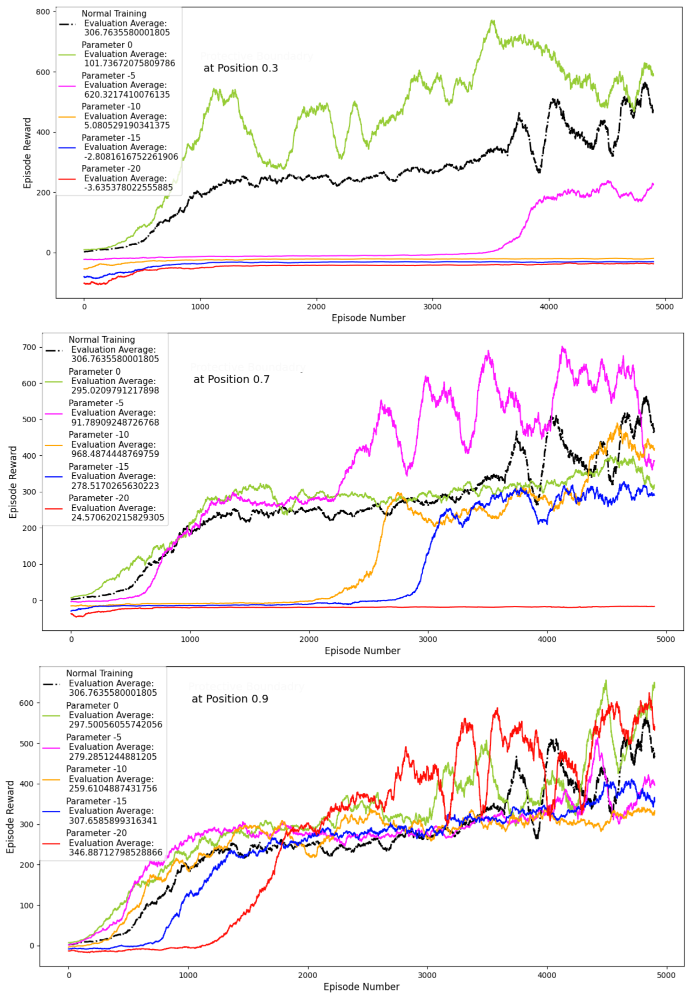


Fig. 10: Walker2D

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] M. Han, L. Zhang, J. Wang, and W. Pan, "Actor-critic reinforcement learning for control with stability guarantee," *IEEE Robotics and Automation Letters*, vol. 5, pp. 6217–6224, 2020.

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

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

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

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

[23] 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.

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

[25] 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.

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

[27] 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.

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

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

[30] 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.

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

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

[33] 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.

[34] 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.

[35] 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.

[36] L. Xie, S. Wang, S. Rosa, A. Markham, and A. Trigoni, “Learning with training wheels: Speeding up training with a simple controller for deep reinforcement learning,” *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6276–6283, 2018.

[37] I. Carlucho, M. D. Paula, S. A. Villar, and G. G. Acosta, “Incremental q-learning strategy for adaptive pid control of mobile robots,” *Expert Syst. Appl.*, vol. 80, pp. 183–199, 2017.

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

[39] 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

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