Coaching with PID Controllers: A Novel Approach for Merging Classic Control with Learning

Zongqiang Pang, Liping Bai

***Abstract*—In this paper, we propose a Proportional Integral Derivative (PID) controller based coaching scheme in order to expidite data collection on critical states and accelerate reinforcement learning (RL). In previous attempts for fusing classical control and RL, training accelerations are brought about with the implicit cost of caps on what is attainable by the RL agents, and high quality controllers are required as a result. We ask if it possible to accelerate RL with even a basic hand-tuned PID controller, which is primitive yet widely used. We draw inspiration from the relationship between atheletes and their coaches. World class atheletes are by definition the top of their field, yet training with coaches who are less capable than them is still an integral part of their skills acquisition process. The job of a coach is not to function as a template to be imitated, but rather to provide conditions for the atheletes to collect critical experiences and that is what we seek to replicate in our approach. Experiments are conducted in Mujoco locomotion environments, specifically the inverted pendulum, inverted double pendulum, hopper, and walker simulations. We conclude from the data that when the coaching structure between PID controller and its respective RL agent is set at its goldilocks spot, agents training can be accelerated, yielding uncompromised training results in the mean time. This is an important proof of concept that controller based coaching can be a novel and effective paradigm for merging classical control with learning and warrants further investigations along this direction. All the code and data can be found at: github.com/BaiLiping/Coaching**

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

I. Introduction

REINFORCEMENT Learning (RL) cyles between interaction with an environment and refinement of the understanding of that environment. 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 trajectory. 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. The focus of current researches is to re-formulate RL into its more effective format.

There are four angles of attack in order to achieve the aforementioned objective. First, researchers who are well versed in statistics and information theories approach the problem by devising ever more sophisticated statistical processes to push the budgeted data collection and information extraction to its limit.[6][7][8][9][10][11][12][13][14][15][16][17] Second, control theoriests who are capable of wielding optimization techniques and mathematical formalism provide a systematic perspective to RL and invent the much needed analytical tools.[18][19][20][21][22][23][24][25]Third, System Indentification (SI) researchers are exploring all the possible configurations to extend existing system models with Neural Networks. [26][27][28][29][30] Fourth, imitation learning, reverse reinforcement learning, and human-in-the-loop teaching aim at preemptively limitsing exploration to the pertinent states and providing guidance to the data collection process. [31][32][33][34][35]

In this paper, we propose an additional angle of attack: classical controllers based coaching. We are not the first group to realize that classical controllers can be used to guide RL agents and accelerate training. [36][37][38]. Yet all the previous works demand high quality controllers to begin with and the improvements brought about by the RL agents are merely icing on the cake. In addition, the controllers inadvertently impose limits on what can be achieved by the RL agents, if a bad controller is chosen, then the RL training process would be hindered rather than expedited. We ask the question, is there a way to take advantage of hand-tuned PID controllers, which might not be high-quality but still captures some of our understanding of the system, in the RL training process? This inquiry lead us to the relationship between atheletes and their coaches.

Professional athletes don't get where they are via trial-and-error. Their skillsets are forged through coaching. Have you ever wondered why top atheletes who represent the best in that sport still need coaching from people who are less capable? The objective of a coach is to facilitate data collection on critical states, therefore the capacity of the coach per se is not essential for training purposes. This provide us with empirical evidence that training acceleration can be achieved with properly constructed coaching scheme, and the demand on controller quality should not be strigent.

In this paper, we hand-tune the PID controllers as the 'coaches'. The performance of the PID controllers is miserable comapare to the respective RL agent as shown by Table I. Yet, even with such bad controllers, when structured properly, training acceleration is still observed, as shown by Table II. The detailed of our implementation and caveats would be detailed in subsequent sections.

In section II we provide the basic understanding of reinforcement learning as seen from control perspective. In section III, we present the results of all our experiments. While the experiments are conducted in Mujoco 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

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

Environment	PID Controller	RL Agent	PID\RL
Inverted Pendulum	240	1000	24.0%
Double Pendulum	1107	9319	11.9%
Hopper	581	989	58.7%
Walker	528	1005	52.5%

TABLE I: PID Controller Vs Agent Score.Performance Comparison. Hand-Tuned PID controllers perform miserably compare to the perepective RL agent.

Environment Name	Target Score	Measure	With PID Coaching	Without Coaching	Percentage Increase
Inverted Pendulum	800	Win Streak	110	160	3.01%
		Average	110	149	2.6%
Double Pendulum	5500	5 Wins	908	1335	32.0%
		Average	866	1323	34.5%
Hopper	800	5 Wins	2073	2851	27.3%
		Average	2085	2781	25.0%
Walker	800	5 Wins	106	160	13%
		Average	108	159	13%

TABLE II: Comparison Between Agents Trained With and Without PID Controller Coaching.Even though the PID controllers are less capable than the eventual RL agent. During the training phase, it is still helpful and can accelerate the RL agent training, as indicated by the number of episodes training required to reach a predetermined target score.

experiments and layout directions for further research.

II. 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." [41]

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++[42], and software packages such as PyTorch[43] and Tensorflow[44] made this task even more trivial.

In order to better introduce RL to control researchers, we built on Benjamin Recht's presentation[45], framed RL in the language of optimization. Reinforcement Learning is an umbrella term, which encompasses distinctive genres of 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

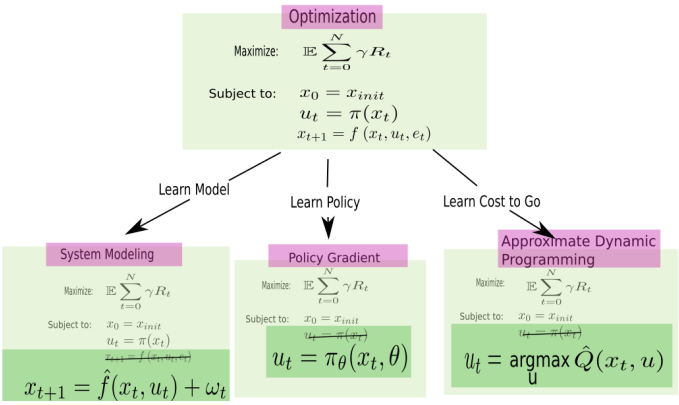


Fig. 1: From Optimization to Learning

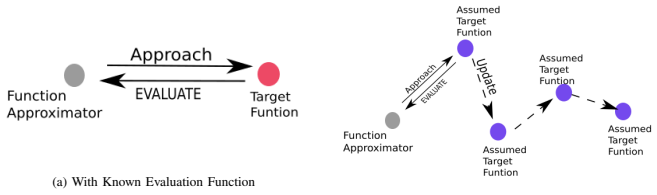
function. If there is an explicit action policy, it is called on-policy learning. Otherwise, the optimal action would be implicitly captured by the Q value function and that would be called off-policy learning instead. Importance sampling allows "limited off-policy learning" capacity, which allows data 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.

A. 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, 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



(a) With Known Evaluation Function

(b) Bootstrap

Fig. 2: Convergence through Bootstrap

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.

B. 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[46]. 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. Experimental Setup

Mujoco physics engine does provide the system models for the simulation, therefore it is possible to compute for more sophiscated controllers.

We implemented our active boundary experiments in OpenAI Gym[47] 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[48]. 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. 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 **??**.

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

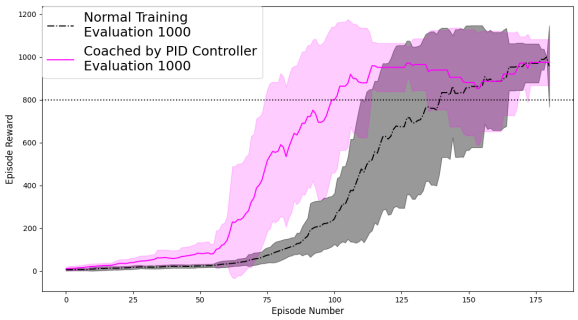


Fig. 3: Inverted Pendulum Coaching Result. Benchmarked against training without coaching, indicated by the black dotted line.

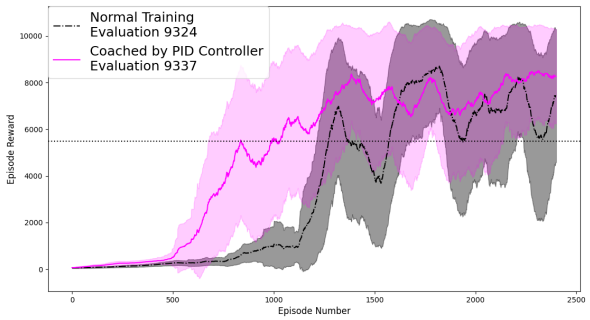


Fig. 4: Double Inverted Pendulum Coaching Result. Benchmarked against training without coaching, indicated by the black dotted line.

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 **??**. The experiment result on inverted pendulum is shown by figure **??**. 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 **??**. 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 **??**. 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.

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 the penalty parameters that is chosen from 0,-3,-5,-7,-10 as shown by Figure **??** 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 **??**. 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

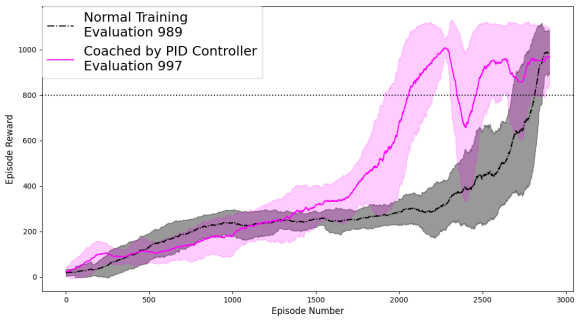


Fig. 5: Hopper Coaching Result. Benchmarked against training without coaching, indicated by the black dotted line.

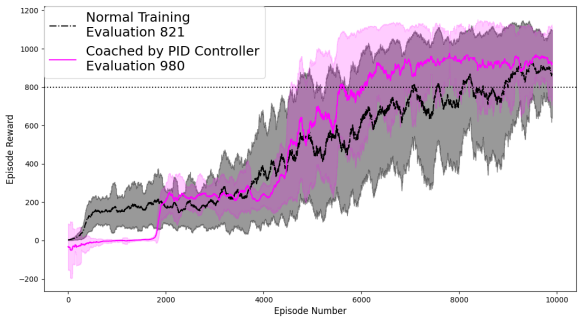


Fig. 6: Hopper Coaching Result. Benchmarked against training without coaching, indicated by the black dotted line.

acceleration is observed when the boundary is set at 0.15 y position.

D. Walker

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 the penalty parameters that is chosen from 0,-3,-5,-7,-10 as shown by Figure **??** 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 **??**. 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.

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 [49]. 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] 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] B. S. Pavse, F. Torabi, J. P. Hanna, G. Warnell, and P. Stone, "Ridm: Reinforced inverse dynamics modeling for learning from a single observed demonstration," *IEEE Robotics and Automation Letters*, vol. 5, pp. 6262–6269, 2020.

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

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

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

[42] 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.

[43] 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.

[44] 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.

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

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

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

[48] 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

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