ROBUST MAML: PRIORITIZATION TASK BUFFER WITH ADAPTIVE LEARNING PROCESS FOR MODEL-AGNOSTIC META-LEARNING

Thanh Nguyen, Tung Luu, Trung Pham, Sanzhar Rakhimkul, Chang D. Yoo

Korea Advanced Institute of Science and Technology (KAIST)

ABSTRACT

Model agnostic meta-learning (MAML) is a popular state-of-the-art meta-learning algorithm that provides good weight initialization of a model given a variety of learning tasks. The model initialized by provided weight can be fine-tuned to an unseen task despite only using a small amount of samples and within a few adaptation steps. MAML is simple and versatile but requires costly learning rate tuning and careful design of the task distribution which affects its scalability and generalization. This paper proposes a more robust MAML based on an adaptive learning scheme and a prioritization task buffer (PTB) referred to as Robust MAML (RMAML) for improving scalability of training process and alleviating the problem of distribution mismatch. RMAML uses gradient-based hyper-parameter optimization to automatically find the optimal learning rate and uses the PTB to gradually adjust training task distribution toward testing task distribution over the course of training. Experimental results on meta reinforcement learning environments demonstrate a substantial performance gain as well as being less sensitive to hyper-parameter choice and robust to distribution mismatch.

Index Terms— meta-learning, reinforcement learning, hyper-parameter optimization, learning to learn

1. INTRODUCTION

Meta-learning, often referred to as learning to learn, has emerged as a potential learning paradigm that can absorb information from tasks and generalize that information to unseen tasks proficiently. It make learning more general: efforts being made to construct task distributions, from which meta-learning algorithms can automatically provide good weight initialization and a set of hyper-parameters. A relatively recent landmark meta-learning algorithm is Model-Agnostic Meta-Learning (MAML) [1] which is a conceptually simple, general algorithm that has shown impressive results over many problems ranging from few-shot learning problems in classification, regression, and reinforcement learning (RL) [2]. MAML trains a model on a task distribution to acquire the optimal weight initialization. The model then can be adapted to an unseen task with few sample and few adaptation steps - often in one step [1].

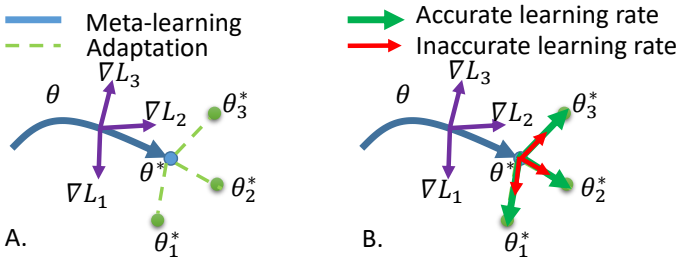

Fig. 1. **A**. shows how MAML learns weight initialization [1]. **B**. shows the effect on the adaptation process of different learner learning rates.

Although MAML has an interesting approach, the generality and simplicity of the algorithm come with two difficulties. First, MAML is considered expensive in terms of computational cost since the model is trained on multiple tasks and requires the computation of the second-order derivative. Moreover, after training, the model is adapted to an unseen task with a few samples and a few adaptation steps that require an accurate learner learning rate. Thus, the time for searching hyper-parameters significantly increases compared to other meta-learning approaches. The difference between the accurate and the inaccurate learning rate of the learner is shown in Fig 1, intuitively. The sensitivity of hyper-parameters to performance can lead to poor scalability. Second, learning the weight initialization requires careful design of task distribution to achieve high performance. Ideally, task representations in latent space should collapse into one mode and the unseen task should conform well to the task distribution. In reality, even with good prior knowledge of the tasks, it is difficult to manually design the perfect task distribution. A slight mismatch in distribution between training tasks and testing tasks distribution can lead to poor generalization in the adaptation process [3]. Thus, uniformly sampling from the training task distribution, in the manner done by MAML, is generally considered to be an inefficient strategy.

This paper proposes Robust MAML (RMAML): an adaptive learning scheme and a prioritization task buffer (PTB) to address the two aforementioned problems. For hyper-

parameter tuning, the learner learning rate can be gradually adjusted to minimize the validation loss over the course of training automatically instead of manual tuning. The learner learning rate can vary among weights or layers of the model to provide more flexibility to the adaptation process without any human effort thanks to the adaptive learning scheme. For distribution mismatch, besides uniformly sampling tasks from task distribution, PTB interferes with the training process by providing additional specified tasks prioritized on validation return. This helps to correct the training task distribution to align with testing task distribution assuming the training task distribution is the unimodal distribution with noise and the testing task distribution aligns with the free noise training task distribution. The validation return is available in meta-learning algorithms that require no extra effort. RMAML shows good performance, helps to stabilize the training process, increases the scalability, and robustness to distribution mismatch.

2. RELATED WORK

Meta-learning involves learning a learning algorithm that can adapt quickly to an unseen task with few samples. Meta learning papers can be classified into the following three approaches: metric-based [4][5][6][7], model-based [8][9][10], and optimization-based [11][12]. This paper directly falls into the category of the optimization-based approach which focuses on the optimization process by customizing the optimizer [13] or finding good weight initialization (MAML [1]). MAML has shown impressive performance but also has some problems. Previously, efforts have been made to reduce computation cost of MAML by using the first-order approximation of gradient [11], raising problems about task distribution sensitivity and using reinforcement learning to correctly choose training tasks [3], or providing adaptation model in a probabilistic way [12]. By contrast, RMAML solves the scalability caused by costly hyper-parameter tuning and task distribution mismatch between training and testing.

The issues discussed in this paper are closely related to the topic hyper-parameter optimization which focuses on finding the best set of hyper-parameters automatically. Many methods have been proposed ranging from naive grid search to more advanced approaches such as using Bayesian [14], model-based [15], reversible learning [16], or hypergradient descent [17]. Leveraging the success of this field but keeping the method lightweight, simple and effective, RMAML chooses gradient-based optimization for hyper-parameter by minimizing the validation loss which is conveniently available in the training process of MAML.

RMAML is performed on Reinforcement Learning (RL) tasks since it is considered the most challenging problem for meta-learning, also known as meta reinforcement learning. The goal is to find a policy that can quickly adapt to an unseen environment from only a few trajectories. Some methods try to solve it by conditioning the policy on a latent representation of the task [18] or using a recurrent neural network [19]. MAML has also shown some achievements utilizing REINFORCE loss [20] for inner adaptation loop and TRPO [21] for outer meta-learning loop.

Algorithm 1 MAML for Reinforcement Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β : step size hyper-parameters
1: Randomly initialize θ
2: Initialize $\alpha = \alpha_0$
3: **while** not done **do**
4: Uniformly sample M tasks $\mathcal{T}_i \quad p(\mathcal{T})$
5: **for all** $\mathcal{T}_i$ **do**
6: Sample K trajectories D^i_{train} using f_θ in $\mathcal{T}_i$
7: Update one-step gradient descent using D^i_{train}
8: $\theta'_i = \theta \quad \alpha \nabla_\theta \mathcal{L}^{train}_{\mathcal{T}_i}(f_\theta)$
9: Sample trajectories D'_i using $f_{\theta'_i}$ in $\mathcal{T}_i$
10: Update initialization weight using each $\mathcal{D}^i_{val}$
11: $\theta \quad \theta \quad \beta \nabla_\theta \sum_{\mathcal{D}_{val}} \mathcal{L}^{val}_{\mathcal{T}_i}\left(f_{\theta'_i}\right)$

3. MAML

MAML is a optimization-based meta learning algorithm that learns an optimal weight initialization for a model given a task distribution. Formally, the model f_θ is parameterized by θ. Given a task generated from task distribution P $\tau_i \sim P(\tau)$ and its associated training and validation datasets are $\left(\mathcal{D}^i_{train}, \mathcal{D}^i_{val}\right)$. The model can be trained by one or more gradient descent steps (adaptation steps) as follows

$$\theta'_i = \theta - \alpha \nabla_\theta \mathcal{L}^{train}_{\tau_i}(f_\theta),$$

where $\mathcal{L}^{train}_{\tau_i}$ is the loss for task τ_i computed using D^i_{train}. Here α is the learning rate of the learner. To achieve stable generalization across P, MAML finds the optimal initialization weight θ^* such that the task-specific fine-tuning achieves low validation loss. The solution can be acquired by minimizing the validation after adapting across τ_i computed using $\mathcal{D}^i_{\text{val}}$:

$$\begin{aligned}\theta^* &= \arg\min_\theta \sum_{\tau_i \sim p(\tau)} \mathcal{L}^{val}_{\tau_i}\left(f_{\theta'_i}\right) \\ &= \arg\min_\theta \sum_{\tau_i \sim p(\tau)} \mathcal{L}^{val}_{\tau_i}\left(f_{\theta - \alpha \nabla_\theta \mathcal{L}^{train}_{\tau_i}(f_\theta)}\right),\end{aligned}$$

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\tau_i \sim p(\tau)} \mathcal{L}^{val}_{\tau_i}\left(f_{\theta - \alpha \nabla_\theta \mathcal{L}^{train}_{\tau_i}(f_\theta)}\right),$$

where β is the meta learning rate and $\mathcal{L}^{val}_{\tau_i}$ is the validation loss for τ_i. For RL meta-learning, $\mathcal{L}^{train}$ is REINFORCE loss [20] and $\mathcal{L}^{val}_{\tau_i}$ is equivalent loss used in TRPO [21]. Each meta update sample M tasks, referred to as meta-batchsize. The pseudo-code of MAML for RL is shown in Algorithm 1.

4. ROBUST MAML

Instead of using a fixed learner learning rate as hyper-parameter, RMAML optimizes the learner learning rate by gradient descent to minimize the evaluation loss. Optionally, the learning rate can vary among weights or layers to provide more flexibility in the adaptation process. In RMAML, one-step gradient of the learner learning rate is given as:

$$\begin{aligned}\frac{\partial \mathcal{L}_{\mathcal{T}_i}^{val}\left(f_{\theta_i'}\right)}{\partial \alpha} &= \frac{\partial \mathcal{L}_{\mathcal{T}_i}^{val}\left(f_{\theta_i'}\right)^T}{\partial \theta_i'}\frac{\partial \theta_i'}{\partial \alpha} \\ &= \nabla_{\theta_i'}\mathcal{L}_{\mathcal{T}_i}^{val}\left(f_{\theta_i'}\right)^T \frac{\partial\left(\theta - \alpha\nabla_\theta \mathcal{L}_{\mathcal{T}_i}^{train}\left(f_\theta\right)\right)}{\partial \alpha} \\ &= \nabla_{\theta_i'}\mathcal{L}_{\mathcal{T}_i}^{val}\left(f_{\theta_i'}\right)^T \cdot \left(-\nabla_\theta \mathcal{L}_{\mathcal{T}_i}^{train}\left(f_\theta\right)\right).\end{aligned}$$

$$\alpha = \alpha + \alpha_0 \nabla_{\theta_i'}\mathcal{L}_{\mathcal{T}_i}^{val}\left(f_{\theta_i'}\right)^T \nabla_\theta \mathcal{L}_{\mathcal{T}_i}^{train}\left(f_\theta\right)$$

Notice that, although α is substituted by another parameter α_0, α_0 is not sensitive to the performance of the algorithm and can be safely set at small value (e.g 1e-2). The meta learning rates can also be adjusted. However, the meta learning rate is not sensitive to the performance compared to the learner learning rate. Therefore, the meta-learning rate is set equal to α_0.

In terms of training task distribution P, MAML uniformly samples task from P (UNIFORM STRATEGY). It performs well with uni-modal distribution but is sensitive to noisy tasks. Intuitively, the noisy tasks, far from the mode of P, can pull the solution away from the optimal initialization weight. Actively training models more on useful tasks yields better convergence. The problem is that P can not be accessed directly. Querying task information consumes computation cost and the information highly depends on the querying RL policy. Wisely using the task information is crucial. RMAML introduces a prioritization task buffer (PTB), denoted by B. PTB adds more useful tasks during training besides uniformly sampling from the task distribution P as follows: at every meta update, all M training tasks are kept in B with corresponding validation returns. In the next iteration, M tasks, required for learner update, are L tasks sampled from B plus $(M - L)$ tasks uniformly sampling from P. After sampling, B is cleared to prepare it to receive new tasks. The L tasks, whose validation return is relatively medium in B, are chosen. L is gradually increased from 0 to MAX_L ($\leq M$) over the course of training (MEDIUM STRATEGY).

Fig 2 shows the behaviour of MAML/RMAML in two type distributions. Ignoring the bias inducing by noisy tasks, the majority of useful tasks will gradually pull the resulting weight towards the optimal weight. PTB helps to increase the rate of training on the useful tasks to eliminate bias. Furthermore, the better the RL policy the more trust-able the buffer

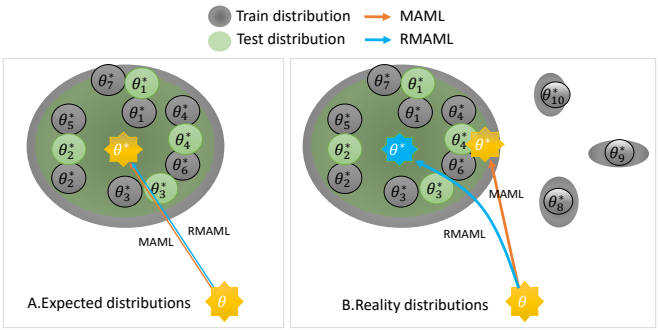

Fig. 2. The demonstration of distribution mismatch between train/test and the corresponding behavior of MAML/RMAML.

since validation return is more distinguishable between useful tasks and noisy tasks. Using a small number of L in the beginning of the training and gradually increase its value showed better performance. Regarding how to choose useful tasks in B, an interesting empirical observation is that the tasks with the lower validation loss (EASY STRATEGY) tend to move the policy to a local minimums, whereas tasks with higher validation loss (HARD STRATEGY) will move policy somewhere far from the optimum. Tasks with medium validation loss (MEDIUM STRATEGY) successfully move the policy closer to the optimum. This strategy is consistent with curriculum learning which proposes training on tasks that are not too easy and also not too hard [22]. The pseudo-code is shown in Algorithm 2.

Algorithm 2 RMAML for Reinforcement Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α_0 step size hyper-parameter
Require: Initialize Priority Task Buffer B
Randomly initialize θ
while not done **do**
  Sample L tasks $\mathcal{T}_B$ B ▷ MEDIUM STRATEGY
  Uniformly Sample $(M$ $L)$ tasks $\mathcal{T}_P$ $p(\mathcal{T})$
  $\mathcal{T}_i = \mathcal{T}_B$ concat $\mathcal{T}_P$
  Empty B
  for all $\mathcal{T}_i$ **do**
    Sample K trajectories D_{train}^i using f_θ in $\mathcal{T}_i$
    Update one-step gradient descent using D_{train}^i
    $\theta_i' = \theta$ $\alpha \, r_\theta L_{\mathcal{T}_i}^{train}\left(f_\theta\right)$
    Sample trajectories D_i' using $f_{\theta_i'}$ in $\mathcal{T}_i$
  Update initialization weight using each D_{val}^i
  θ θ $\alpha_0 \, r_\theta \sum_{\mathcal{D}_{val}} L_{\mathcal{T}_i}^{val}\left(f_{\theta_i'}\right)$
  $\alpha = \alpha + \alpha_0 \, r_{\theta_i'} L_{\mathcal{T}_i}^{val}\left(f_{\theta_i'}\right)^T$ $r_\theta L_{\mathcal{T}_i}^{train}\left(f_\theta\right)$
  Store all $\mathcal{T}_i, L_{\mathcal{T}_i}^{val}\left(f_{\theta_i'}\right)$! B

5. EXPERIMENTS

Sampling Strategy Investigation. Our team designed a environment called 2D REACHING which consists of 300 tasks that are drawn from a mixture of three independent normal distributions. Each task corresponds to a 2D point which a RL policy is required to reach. The largest normal distribution contains 200 points (assuming it matches the testing distribution). Each remaining normal distribution contains 50 tasks playing the role as noise tasks. Assume we have a perfect MAML algorithm that can move the agent toward the given task proportional to the distance between them. The weight of the agent is the current position of the agent. Prioritization scores are the distance between tasks and the updated agent position. Different PBT strategy are investigated: EASY, HARD, MEDIUM, UNIFORM. The goal is to make MAML produce the weight near the center's largest normal distribution.

The results in Fig 3 show that the MEDIUM strategy allows the robot to move to the desired destination without being stuck in local minima.

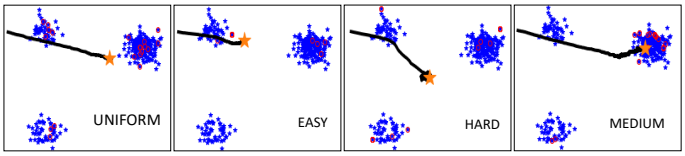

Fig. 3. Different sampling strategies performed on the 2D REACHING environment. UNIFORM: random sampling EASY: use PTB prioritizing easy tasks, HARD: use PTB prioritizing hard tasks, MEDIUM: use PTB prioritizing medium tasks.

Reinforcement Learning Tasks. To demonstrate the effectiveness, RMAML is evaluated on high dimensional locomotion tasks simulated in MuJoCo [23]. For fare comparison, the same setup in MAML [1] is applied. There are two tasks: Cheetah Velocity (VEL), Cheetah Direction (DIR). In VEL, a Cheetah robot must run at a particular velocity, chosen uniformly at a random value between 0.0 and 2.0. In DIR, the Cheetah robot must run in a particular, randomly chosen direction (forward/backward). The hyper-parameters are used in the same way mentioned in MAML [1].

For RMAML specific hyper-parameter, we set $\alpha_0 = 0.01$, L is gradually increased up to 1/4 of the meta-batch size. For implementation, to reduce the wall time, RMANL is implemented using distributed training which is currently popular in the RL field [24]. For testing, 40 tasks are sampled randomly. The model is initialized by RMAML and evaluated over 40 tasks (roll out 20 trajectories for each task). The average return among trajectories and tasks is reported as step 0. Then, the model performs one step adaptation with gradient descent, rollout and the average return is reported as step 1. Table 1 shows the average test return from different algorithms: RMAM, MAML+ (our MAML implementation with distributed training), MAML[1] and Pretrain[1]. The result shows that RMAML consistently outperforms MAML+ in both environments and outperform the original MAML in the VEL. To verify the robustness of RMAML with distribution mismatch, the Cheetah Velocity is customized to be Noise Cheetah Velocity (NoiseVEL). NoiseVEL adds 20% noise tasks chosen uniformly at random between 3.0 and 4.0 during training. During test time we stop adding noise tasks and evaluate the algorithm as mentioned above.

Task	Step	Pretrain[1]	MAML[1]	MAML+ [1]	RMAML
VEL	0	-158.0	-125.0	-60.1	**-58.0**
	1	-137.0	-79.0	-41.5	**-31.2**
DIR	0	-40.5	-50.7	30.3	**18.2**
	1	-38.3	**293.2**	215.7	272.9

Table 1. Result of the RL locomotion tasks featuring the average test return on Half Cheetah Velocity [VEL], Half Cheetah Direction [DIR], 2D Navigation [2D] with 0 and 1 adaptation step.

Phase	Step	MAML+[1]	RMAML
Train (Noise)	0	-99.1	**-84.3**
	1	-71.7	**-60.4**
Test	0	-60.3	**-42.0**
	1	-54.9	**-31.8**

Table 2. Noise Cheetah Velocity Result featuring average test return during the training phase and testing phases with and without noise.

The results in Table 2 shows that RMAML outperforms MAML+ on both training and testing as well as reach to near -31.2 similar to training on VEL which has no noise.

6. CONCLUSION

This paper presents Robust MAML, a prioritization task buffer with an adaptive learning process for model-agnostic meta-learning. RMAML substantially reduces hyper-parameter tuning time and it is robust to distribution mismatch. This makes RMAML suitable and effective to scale for a variety of problems. RMAML shows consistent results that outperforms the original MAML locomotion meta-RL benchmarks. This paper treats the problem of the distribution mismatch between training task distribution and testing task distribution as unimodal distribution with noise and solves it. Future research should pay attention to more complex distributions such as multimodality distributions.

[1]MAML reimplementation using distributed training

7. REFERENCES

[1] Chelsea Finn, Pieter Abbeel, and Sergey Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1126–1135.

[2] Antreas Antoniou, Harrison Edwards, and Amos Storkey, "How to train your maml," *arXiv preprint arXiv:1810.09502*, 2018.

[3] Bhairav Mehta, Tristan Deleu, Sharath Chandra Raparthy, Chris J Pal, and Liam Paull, "Curriculum in gradient-based meta-reinforcement learning," *arXiv preprint arXiv:2002.07956*, 2020.

[4] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*. Lille, 2015, vol. 2.

[5] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al., "Matching networks for one shot learning," in *Advances in neural information processing systems*, 2016, pp. 3630–3638.

[6] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales, "Learning to compare: Relation network for few-shot learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1199–1208.

[7] Jake Snell, Kevin Swersky, and Richard Zemel, "Prototypical networks for few-shot learning," in *Advances in neural information processing systems*, 2017, pp. 4077–4087.

[8] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap, "Meta-learning with memory-augmented neural networks," in *International conference on machine learning*, 2016, pp. 1842–1850.

[9] Alex Graves, Greg Wayne, and Ivo Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[10] Tsendsuren Munkhdalai and Hong Yu, "Meta networks," in *Proceedings of the 34th ICML-Volume 70*. JMLR. org, 2017, pp. 2554–2563.

[11] Alex Nichol, Joshua Achiam, and John Schulman, "On first-order meta-learning algorithms," *arXiv preprint arXiv:1803.02999*, 2018.

[12] Chelsea Finn, Kelvin Xu, and Sergey Levine, "Probabilistic model-agnostic meta-learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 9516–9527.

[13] Sachin Ravi and Hugo Larochelle, "Optimization as a model for few-shot learning," *arXiv preprint*, 2016.

[14] Jasper Snoek, Hugo Larochelle, and Ryan P Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems*, 2012, pp. 2951–2959.

[15] James Bergstra, Daniel Yamins, and David Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *ICML*, 2013, pp. 115–123.

[16] Dougal Maclaurin, David Duvenaud, and Ryan Adams, "Gradient-based hyperparameter optimization through reversible learning," in *ICML*, 2015, pp. 2113–2122.

[17] Atilim Gunes Baydin, Robert Cornish, David Martinez Rubio, Mark Schmidt, and Frank Wood, "Online learning rate adaptation with hypergradient descent," *arXiv preprint arXiv:1703.04782*, 2017.

[18] Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen, "Efficient off-policy meta-reinforcement learning via probabilistic context variables," in *International conference on machine learning*, 2019, pp. 5331–5340.

[19] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel, "Rl 2: Fast reinforcement learning via slow reinforcement learning," *arXiv preprint arXiv:1611.02779*, 2016.

[20] Ronald J Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[21] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz, "Trust region policy optimization," in *ICML*, 2015, pp. 1889–1897.

[22] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.

[23] Emanuel Todorov, Tom Erez, and Yuval Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[24] Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva Tb, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap, "Distributed distributional deterministic policy gradients," *arXiv preprint arXiv:1804.08617*, 2018.