FINAL REPORT

ITCS 6880 – Individual Study

Suhail Lala
slala@uncc.edu

FINAL REPORT

Submitted by
Suhail Lala

Under the guidance of

Dr. Minwoo Lee

The University of North Carolina at Charlotte
Charlotte, NC 28223
May 2021

CONTENTS

1) BACKGROUND 3

2) META LEARNING 4

- Approaches to Meta Learning

3) META REINFORCEMENT LEARNING 5

- Key Components
- Approaches for Meta-RL

4) MODEL-AGNOSTIC META LEARNING 6

- Performance
- Critical Analysis

5) EVOLVED POLICY GRADIENTS 9

- Performance
- Critical Analysis

6) FEDERATED & DISTRIBUTED LEARNING 12

- How it works
- Types

7) AVERAGE REGRET-UPPER-BOUND ANALYSIS 13

- Advantages
- Performance
- Critical Analysis

8) MULTI-AGENT ONLINE META LEARNING 15

- Performance
- Critical Analysis

9) SURVEY OF OTHER DISTRIBUTED LEARNING TECHNIQUES 17

- Decentralized Actor-Critic
- Distributed VD-RL
- Multi-Agent Meta-RL

10) REFERENCES 18

1. INTRODUCTION

Reinforcement Learning (RL) is an area of machine learning concerned with how intelligent agents ought to take actions in an environment to maximize the notion of cumulative reward. It is one of three basic machine learning paradigms, alongside supervised and unsupervised learning.

Like the other two paradigms, a reinforcement learning model requires training with a large number of samples. On the other hand, a meta-learning model trained on a particular task can be used on a different task resulting in sample efficiency [1]. Moreover, machine learning models are good only for the tasks or environments that they are trained on. A strong meta-learning model, on the other hand, should be able to generalize to new tasks or conditions that have never been experienced during training. This adaptation process takes place at test with minimal exposure to new configurations.

Meta Learning when applied to Reinforcement Learning is known as Meta Reinforcement Learning (Meta-RL) [2]. Meta-RL techniques typically meta-learn parameters, hyperparameters, loss functions, or exploration strategies. Model-Agnostic Meta-Learning (MAML) [3] is one such meta-learning technique. It is a general optimization algorithm that focuses on meta-learning the parameters and can be used with any model that learns via gradient descent. On the other hand, another technique – Evolved Policy Gradients (EPG) [4] – is Meta-RL-specific and aims to meta-learn the hyper parameters.

Another technique for decreasing training time is to train a model across multiple servers. This machine learning technique, called Distributed Learning, seeks to take advantage of parallelism by spreading the model across different servers [5]. It is distinct from Federated Learning, which aims to train on heterogenous datasets distributed across edge devices. MAOML [6] is one technique for Distributed Learning that extends ARUBA [7] to multi-agent environments. Other Distributed Learning techniques include decentralized actor-critic [8], distributed VD-RL [9], and multi-agent meta-RL [10].

2. META LEARNING

A well-trained machine learning model frequently requires a large number of training samples. Humans, on the other hand, acquire new concepts and skills much more quickly and efficiently. For instance, children who have only seen cats and birds a few times can quickly distinguish them. Meta learning aims to design models with similar properties — rapid learning of new concepts and skills using only a small number of training examples.

A good meta-learning model can adapt or generalize to novel tasks and environments not encountered during training. The adaptation process, which is essentially a mini-learning session, occurs during test but with only a brief exposure to the novel task configurations. Once adapted, the model would eventually be able to complete new tasks. Therefore, meta-learning is also called learning to learn [1].

Typically, a meta-learning model is trained on a variety of different learning tasks and optimized for optimal performance across a range of tasks, including potentially unknown tasks. Each task corresponds to a dataset D, which contains both feature vectors and true labels. The optimal model parameters are:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{\mathcal{D}\sim p(\mathcal{D})}[\mathcal{L}_\theta(\mathcal{D})]$$ [1]

In this context, a dataset is regarded as a single data sample.

Approaches to meta-learning:

- Model-based: Models update parameters rapidly with few training steps. Examples: Memory-Augmented Neural Networks (MANN), Meta Networks.

- Metric-based: Compares the similarity between two data samples by calculating the distance between them. Examples: Convolutional Siamese Neural Network, Matching Networks, Relation Network, Prototypical Networks.

- Optimization-based: Adjusts the optimization algorithm so that the model can be trained using a small number of examples. Examples: Meta-Learner, Model-Agnostic Meta-Learning (MAML), Reptile.

3. META REINFORCEMENT LEARNING

Meta Reinforcement Learning is to do Meta Learning in the field of Reinforcement Learning [2]. Its objective is to develop agents capable of performing previously unseen tasks quickly and efficiently. Typically, the train and test tasks are distinct but belong to the same problem family. For example, an agent can be trained on a maze having a specific layout and deployed to a maze with a different layout – the agent adapting to the new layout.

Wang et al. (2016) and Duan et al. (2017) concurrently proposed the concept of Meta-RL in its current form [2]. Although their concepts differ slightly, in both cases, a Meta-RL model is trained over a distribution of Markov Decision Processes (MDPs) and is capable of quickly learning to solve a new task during test.

Meta-RL imposes certain types of inductive biases from the task distribution. The term "inductive bias" refers to a set of assumptions that the learner makes to predict outputs given unknown inputs. As a general rule in machine learning, a learning algorithm with a low inductive bias can handle a greater range of variance but is typically less sample-efficient [11]. Thus, narrowing down the hypotheses with stronger inductive bias aids in increasing learning speed.

Key Components:

- A Model with Memory: Recurrent neural networks maintain a hidden state. By updating the hidden state during rollouts, they can acquire and memorize knowledge about the current task. Thus, Meta-RL would be ineffective without memory.

- Meta-learning algorithm: Meta-RL requires a meta-learning algorithm to solve an unknown task during test time.

- A Distribution of MDPs: While the agent is exposed to a variety of environments and tasks during training, it must also acquire the ability to adapt to a variety of MDPs.

Approaches for Meta-RL:

- Meta-learning Parameters: Updates model parameters to ensure that the model performs well on new tasks. Examples: Model-Agnostic Meta-Learning (MAML); Reptile.

- Meta-learning Hyperparameters: Tunes and learns hyperparameters as an agent interacts with its environment. Example: Meta-Gradient RL.

- Meta-learning the Loss Function: The loss function's parameters are evolved so that an agent can achieve higher returns. Example: Evolved Policy Gradients.

- Meta-learning the Exploration Strategies: Learn structured action noise from prior experience for better and more effective exploration. Example: MAESN.

4. MODEL-AGNOSTIC META-LEARNING (MAML)

MAML [3] is a relatively general optimization algorithm that can be used with any model that learns via gradient descent. The central idea is to train the model's initial parameters in such a way that the model performs optimally on a new task after the parameters have been updated via one or more gradient steps computed using a small amount of data from the new task.

The process of training a model's parameters such that a few gradient steps, or even a single gradient step, can produce good results on a new task can be viewed as building an internal representation that is broadly suitable for a variety of tasks. If the internal representation is suitable for a wide variety of tasks, slightly fine-tuning the parameters can produce good results [3]. From a dynamical systems perspective, MAML's learning process can be thought of as optimizing the sensitivity of the loss functions of new tasks to parameter changes: when the sensitivity is high, small parameter changes can result in significant improvements in task loss [3].

Assume our model is f_θ with parameters θ. When adapting to a new task τ_i, one or more gradient descent steps can be used to update the model parameters. Model parameters are trained by optimizing the performance of $f_{\theta'}$ with respect to θ across tasks sampled from p(τ). More concretely, the meta-objective is:

$$\min_{\theta} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i}) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)})$$

[1]

SGD is used to perform meta-optimization across tasks, updating the parameters θ as follows:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i})$$

[1]

Performance:

Fast Learning:

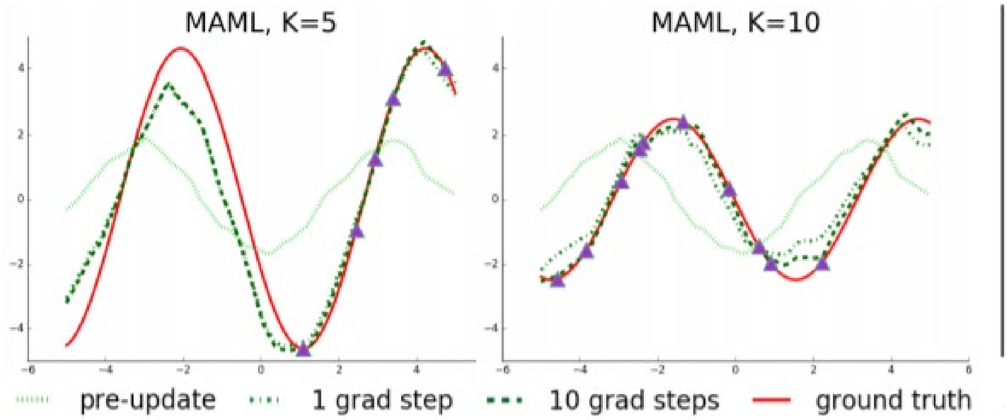


Figure 1. Few-shot adaptation for periodic structure of sine waves. [3]

The above graph shows that MAML's model adapts rapidly with only five data points, denoted by purple triangles, whereas the model pretrained using standard supervised learning on all tasks is incapable of adapting adequately with so few data points. This implies that MAML optimizes the parameters to make them amenable to rapid adaptation.

Continuous Improvement:

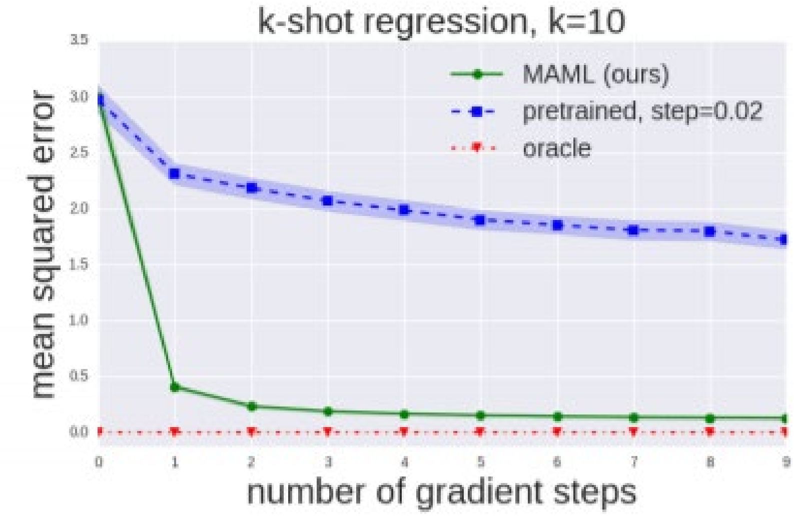


Figure 2. Quantitative sinusoid regression results showing the learning curve at meta test-time. [3]

MAML continues to improve with additional gradient steps without overfitting to the extremely small dataset used for meta-testing, resulting in a loss that is significantly less than that of the baseline fine-tuning approach.

Reinforcement Learning Performance:

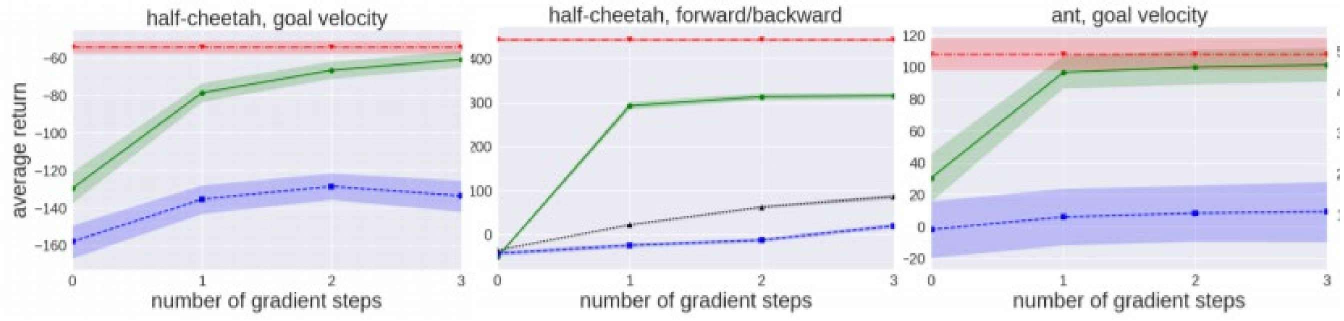


Figure 3. Reinforcement learning results for the half-cheetah and ant locomotion tasks. [3]

The above results demonstrate that MAML can adapt to new goal velocities and directions significantly faster than either conventional pretraining or random initialization, achieving good performance in as few as two or three gradient steps.

Critical Analysis:

MAML is one of the earliest models in Meta-RL. It’s simplicity and applicability to different Machine Learning domains and tasks makes it a popular Meta-RL model. However, the paper focuses mainly on MAML with regards to few-shot learning. Reinforcement Learning is fundamentally different from Supervised Learning. It is difficult to say whether MAML can be applied in the same way as Supervised Learning to a broad range of Reinforcement Learning tasks.

In Reinforcement Learning, there is a tendency to overfit the data [12]. Although the paper addresses the question of overfitting, it is primarily concerned with Supervised Learning. It makes no attempt to determine whether similar generalization results hold true for Reinforcement Learning tasks.

MAML demonstrates a high capacity for rapid learning. However, the paper makes no attempt to explain why such high performance is possible. Recent work demonstrated that the performance of MAML is due to feature reuse rather than rapid learning [13]. MAML's inability to perform well on out-of-distribution tasks may be a result of this deficiency in rapid learning.

Lastly, MAML is compatible with any model that learns by gradient descent [3]. As a result, it may not be suitable for non-convex optimization problems.

5. EVOLVED POLICY GRADIENTS (EPG)

Humans have well-developed internal reward functions because of prior experience or biological evolution. On the other hand, RL agents begin in a blank state and are guided in their initial behavior by external reward signals. EPG [4] advances the goal of creating agents that are not blank slates but rather understand what it means to make progress on a new task based on prior experience of making progress on similar tasks.

EPG's goal is to evolve a differentiable loss function such that an agent can achieve high rewards by optimizing its policy to minimize this loss. This loss function is highly adaptive to the environment and agent history, which results in increased learning speed and the possibility of learning without external rewards. It internalizes an agent's notion of progress towards completing a task and is parametrized using temporal convolutions applied to the agent's experience.

Two optimization loops comprise the method. The inner loop is where an agent learns to solve a task by sampling from a distribution over a family of tasks. By minimizing the loss function provided by the outer loop, an agent learns to solve this task. The outer loop adjusts the parameters of the loss function through Evolutionary Strategies [14] to maximize the final returns obtained following inner loop learning.

The objective of the inner loop optimization problem is to minimize the loss L_ϕ as a function of the agent's policy π_θ:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\tau\sim\mathcal{M},\pi_{\boldsymbol{\theta}}}[L_{\boldsymbol{\phi}}(\pi_{\boldsymbol{\theta}},\tau)].$$

[4]

The outer loop's goal is to learn L_ϕ in such a way that an agent's policy π_θ trained with the loss function produces high expected returns in the MDP distribution:

$$\boldsymbol{\phi}^* = \arg\max_{\boldsymbol{\phi}} \mathbb{E}_{\mathcal{M}\sim p(\mathcal{M})}\mathbb{E}_{\tau\sim\mathcal{M},\pi_{\boldsymbol{\theta}^*}}[R_\tau].$$

[4]

Performance:

Rapid Learning:

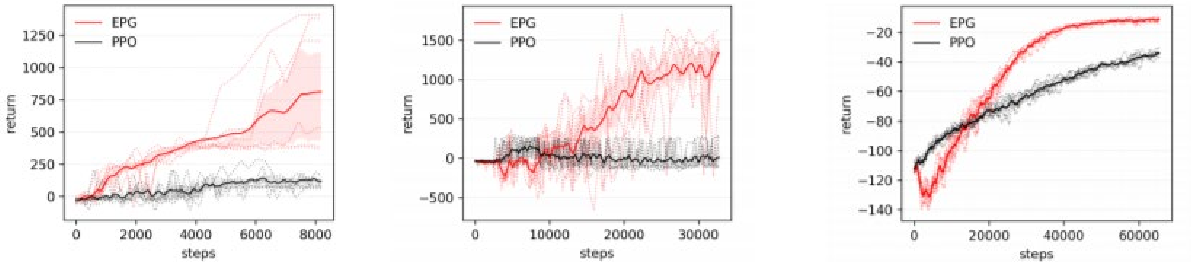


Figure 4. RandomHopper, RandomWalker and RandomReacher test-time training. [4]

The plots illustrate the episodic return in relation to the number of environment steps completed thus far. In all experiments, EPG agents learn faster and obtain higher returns than PPO agents.

The PPO agent learns in these experiments by observing reward signals, whereas the EPG agent does not. At meta-test, EPG does not require observation of rewards, as any piece of information encountered by the agent serves as an input to the EPG loss function. It makes no difference how the agent determines which task to solve within the distribution, as long as this determination is made via observations or rewards [4]. This setting demonstrates the utility of EPG in situations where rewards are only available during meta-training, such as when a system is trained in simulation but deployed in the real world where reward signals are difficult to measure.

Exploratory Behavior:

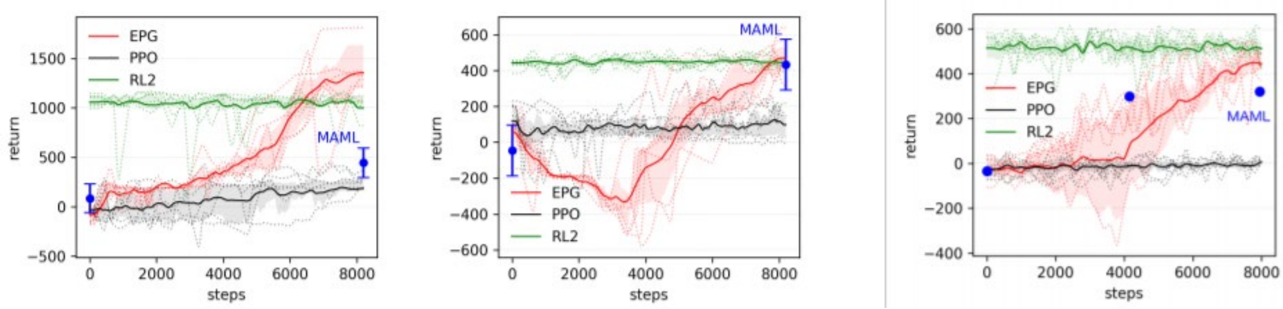


Figure 5. DirectionalHopper and DirectionalHalfCheetah environments. [4]

As seen above, all three methods achieve comparable performance after 8000 timesteps of experience. By simultaneously learning the algorithm and the policy initialization, RL^2 outperforms both MAML and EPG. This, however, comes at the expense of generalization power [4]. Similarly, in most instances, PPO agents stagnate in their learning, whereas EPG is able to train agents that exhibit exploratory behavior.

Out-of-Distribution Tasks:

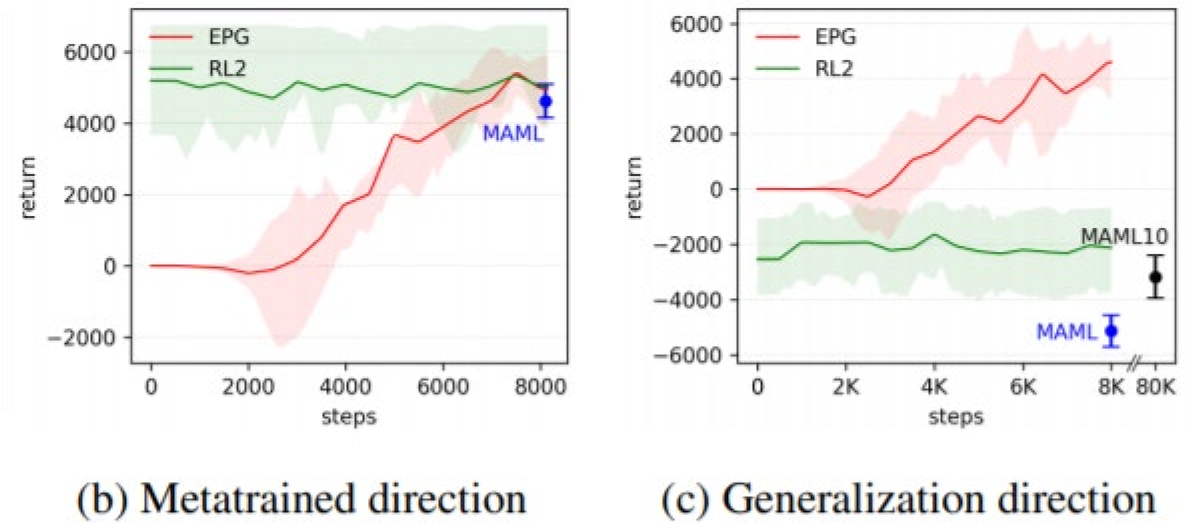


Figure 6. Generalization in GoalAnt. [4]

Both RL^2 and MAML overfit to the task distribution in this case. RL^2 has a flat curve and appears to have overfit completely. It has not been able to learn a general learning algorithm. While MAML does make progress towards improving the returns with multiple gradient steps, it still learns at a much slower rate. On the other hand, EPG evolves a loss function that trains agents to reach previously unseen goals quickly. This demonstrates the generalization properties of learning a loss function decoupled from the policy.

Critical Analysis:

EPG faces some practical challenges, primarily related to the high computational resource requirements and relatively low data efficiency required to learn a good loss function. Since the loss function updates must be performed sequentially, the method's current implementation limits the amount of parallel processing possible. Similarly, because Evolutionary Strategy is a trial-and-error process, the data requirements for EPG are higher than those of other Meta-Learning algorithms.

Additionally, we can currently train an EPG loss to be effective for a single small family of tasks at a time. For instance, instructing an ant to walk left and right. Such EPG loss is unlikely to be effective on a completely different task [4]. For instance, playing Space Invaders. In comparison, standard RL losses do possess this degree of generality – the same loss function can be used to acquire a vast array of skills.

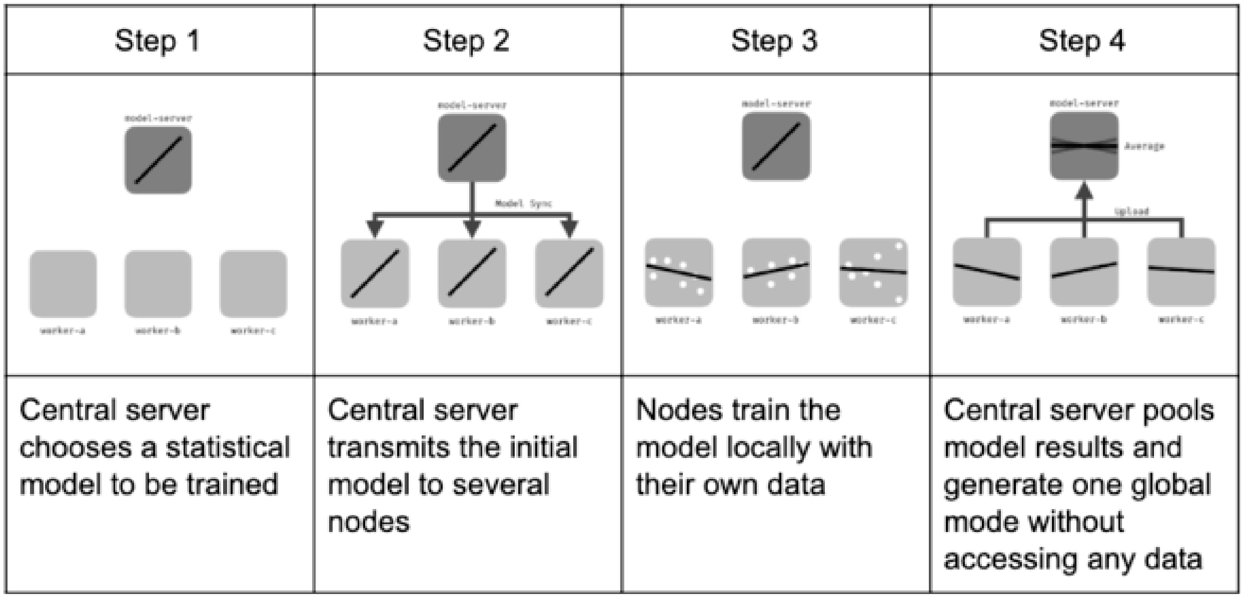
Finally, because Evolutionary Strategy is a black box technique [14], it complicates the understanding and implementation of EPG.

6. FEDERATED & DISTRIBUTED LEARNING

Federated and distributed learning are machine learning techniques that train an algorithm across multiple decentralized edge devices or servers that store local data samples without requiring them to be exchanged. This approach contrasts with traditional centralized machine learning techniques, in which all local datasets are uploaded to a single server.

Federated and distributed learning are distinct in their objectives. While distributed learning focuses on parallelizing computing power, federated learning is concerned with training heterogeneous datasets [5]. Due to the differences in their objectives, distributed learning requires datasets to be independent and identically distributed, whereas federated learning does not make such assumptions. Federated learning is mainly used in smartphones, the Internet of Things, and other heterogeneous data-contained devices, whereas distributed learning is primarily used in data centers [15].

How it works:



Step 1	Step 2	Step 3	Step 4
Central server chooses a statistical model to be trained	Central server transmits the initial model to several nodes	Nodes train the model locally with their own data	Central server pools model results and generate one global mode without accessing any data

Figure 7. Federated learning general process in central orchestrator setup. [16]

Types:

- Centralized: A central server orchestrates the various steps of the algorithms to coordinate the learning process across all participating nodes.
- Decentralized: Nodes coordinate with one another to obtain the global model.
- Heterogeneous: Training heterogeneous local models with dynamically varying computational complexity while maintaining a single global inference model.

7. AVERAGE REGRET-UPPER-BOUND ANALYSIS (ARUBA)

Gradient-based Meta Learning algorithms make use of a very limited, but natural, concept of task similarity – proximity to a single fixed point in the parameter space. On the other hand, ARUBA [7] enables the derivation of machine learning algorithms with a much more sophisticated structure. ARUBA treats meta-learning as the online learning of a sequence of losses, each of which limits the regret associated with a single task [7]. These bounds frequently have convenient functional forms that are (a) smooth and convex, allowing them to be used in conjunction with the existing online convex optimization literature, and (b) strongly dependent on both the task data and the meta-initialization, effectively encoding task similarity mathematically.

The learner has access to a collection of learning algorithms that are parameterized by x ∈ X. On each task t, the meta-learner selects x_t ∈ X, executes the associated algorithm, and experiences regret:

$$\mathbf{R}_t(x_t) = \sum_{i=1}^{m_t} \ell_{t,i}(\theta_{t,i}) - \min_\theta \sum_{i=1}^{m_t} \breve{\ell}_{t,i}(\theta) \text{ [7]}$$

ARUBA examines the performance of the meta-learner by examining the online learning of a sequence of regret-upper-bounds $U_t(x_t) \geq R_t(x_t)$. More precisely, it examines the performance by binding the average regret-upper-bound:

$$\bar{\mathbf{U}}_T = \frac{1}{T} \sum_{t=1}^{T} \mathbf{U}_t(x_t) \text{ [7]}$$

Advantages:

- ARUBA produces a straightforward gradient-based algorithm that eliminates the need to guess similarity by learning it on-the-fly.
- ARUBA reduces the meta-learning problem in dynamic environments to a dynamic regret minimization problem that can be solved directly using online algorithms.
- ARUBA takes advantage of powerful results in online-to-batch conversion to derive new bounds on the transfer risk when using gradient-based meta-learning for statistical learning-to-learn.

Performance:

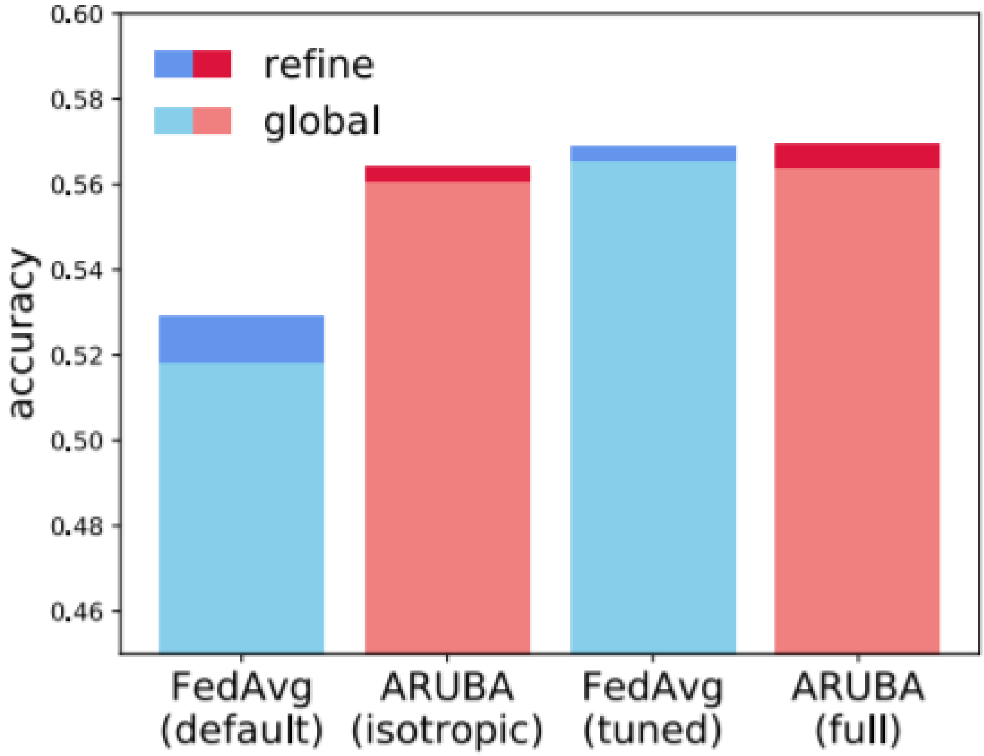


Figure 8. Next-character prediction performance for recurrent networks trained on the Shakespeare dataset. [7]

As illustrated in Figure 8, ARUBA outperforms non-tuned FedAvg and is comparable to FedAvg with a tuned learning rate schedule. Unlike both FedAvg methods, neither of the two ARUBA methods requires rate tuning during model refinement. This decreased need for hyperparameter optimization is critical in federated environments, where user data access is extremely limited. Additionally, because isotropic ARUBA communicates only scalars, its overhead is negligible [7].

Critical Analysis:

ARUBA deals with Single Agent Online Meta-Learning. It might not be appropriate for use in a multi-agent environment. Additionally, it considers online meta-learning to be a convex optimization problem. As a result, it may not be appropriate for non-convex problems.

8. MULTI AGENT ONLINE META LEARNING (MAOML)

Despite online meta-learning's superior fast learning performance, a single agent must learn over a large number of tasks to develop a good meta-model for within-task fast adaptation, which is referred to as the cold-start problem. In a multi-agent network, however, the learning tasks performed by different agents in the same environment frequently share some model similarity. MAOML [6] addresses the cold-start problem by leveraging this task similarity.

MAOML is a framework for multi-agent online meta-learning that is implemented as a nested two-level online convex optimization (OCO) problem. It is a distributed online gradient descent algorithm with gradient tracking, in which each agent tracks the global gradient with only one communication step with its neighbors per iteration.

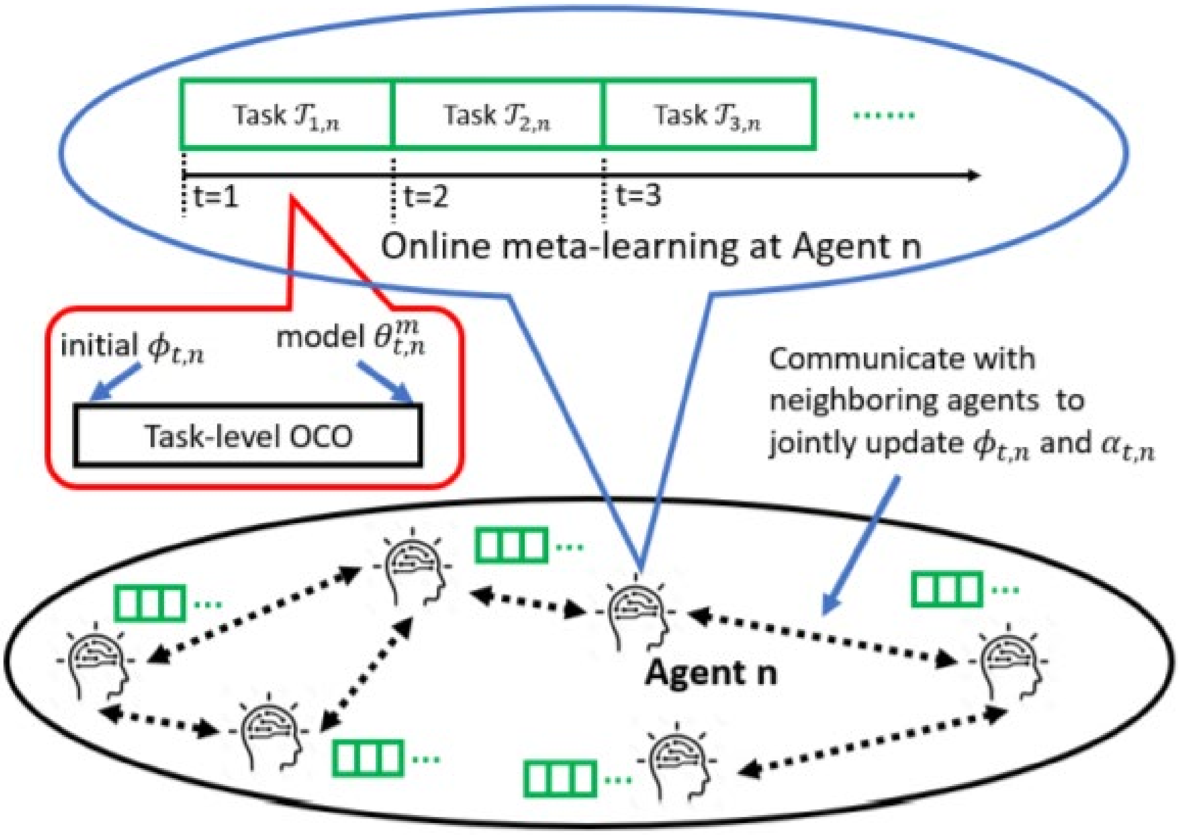
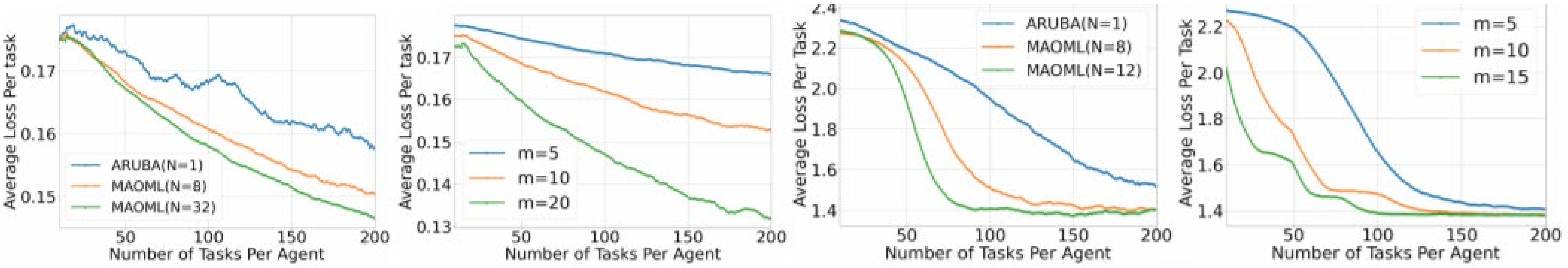


Figure 9. Framework of multi-agent online meta-learning. [6]

MAOML assigns each agent in the multi-agent network a sequence of online learning tasks and shares the learned model knowledge ($\phi_{t,n}$ and a $_{t,n}$) with its neighbors to facilitate the learning of new tasks at time t.

Performance:



(a) Performance comparison for 5-way 10-shot MNIST.

(b) Impact of m on MAOML for 5-way 10-shot MNIST.

(c) Performance comparison for 5-way 5-shot Omniglot.

(d) Impact of m on MAOML for 5-way 5-shot Omniglot.

Figure 10. Performance evaluations of MAOML on MNIST and Omniglot. [6]

MAOML clearly outperforms ARUBA in both convex and non-convex setups by leveraging task similarity across multiple agents via limited communication. In comparison to ARUBA, MAOML rapidly acquires good model priors and performs significantly better after each agent has completed the same number of tasks.

Critical Analysis:

MAOML requires some similarity between the learning tasks of different agents. It may not be appropriate in situations where agents in the same environment are performing wildly dissimilar tasks.

Additionally, MAOML requires that all agents receive tasks at the same rate, which may not always be the case.

9. SURVEY OF OTHER DISTRIBUTED LEARNING TECHNIQUES

Decentralized Actor-Critic [8]**:**

Zhang et al. (2018) proposes two algorithms for fully decentralized multi-agent reinforcement learning. In both these algorithms, each agent performs the actor step independently without the need to infer the policies of the other agents. For the critic step, there is a consensus update via network communication. The proposed algorithms are incremental and can be used in an online environment.

However, because these algorithms are decentralized, they have a higher communication overhead than centralized algorithms. Additionally, there may be numerous communication links between nodes, resulting in an increase in communication overhead when compared to MAOML with limited communication.

Distributed VD-RL [9]**:**

This algorithm enables Drone Base Stations (DBS) to learn their trajectories dynamically while also generalizing their learning to previously unseen environments. Due to the inability of DBS to train value functions independently, Distributed VD-RL decomposes the value for training across all DBS. Additionally, Distributed VD-RL is well suited to solving non-convex optimization problems.

However, because Distributed VD-RL is restricted to the problem of DBS trajectory design, additional experiments are required to ascertain its applicability to other use cases. Additionally, it considers a cooperative setting - agents working together to accomplish a common goal. It makes no mention of a competitive environment in which agents face off.

Multi-Agent Meta-RL [10]**:**

MAMRL's objective is to increase the efficiency of deep reinforcement learning in multi-agent packet routing in dynamic network environments. It is a decentralized algorithm; the router requires only local data: its own and that of its neighbors. Additionally, MAMRL is capable of adapting to more complex topologies with fewer episodes.

However, the paper compares MAMRL to the Shortest Path Algorithm. A more accurate assessment of MAMRL's performance would be to compare it to the performance of other meta-learning algorithms.

References

[1] L. Weng, "Meta-Learning: Learning to Learn Fast," 30 Nov 2018. [Online]. Available: https://lilianweng.github.io/lil-log/2018/11/30/meta-learning.html. [Accessed 06 May 2021].

[2] L. Weng, "Meta Reinforcement Learning," 23 June 2019. [Online]. Available: https://lilianweng.github.io/lil-log/2019/06/23/meta-reinforcement-learning.html. [Accessed 06 May 2021].

[3] C. Finn, P. Abbeel and S. Levine, "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks," *arXiv:1703.03400,* 2017.

[4] R. Houthooft, R. Y. Chen, P. Isola, B. C. Stadie, F. Wolsk, J. H. and P. Abbeel, "Evolved Policy Gradients," *arXiv:1802.04821,* 2018.

[5] J. Konečný, B. McMahan and D. Ramage, "Federated Optimization: Distributed Optimization Beyond the Datacenter," *arXiv:1511.03575,* 2015.

[6] S. Lin, M. Dedeoglu and J. Zhang, "Accelerating Distributed Online Meta-Learning via Multi-Agent Collaboration under Limited Communication," *arXiv:2012.08660,* 2020.

[7] M. Khodak, M.-F. Balcan and A. Talwalkar, "Adaptive Gradient-Based Meta-Learning Methods," *arXiv:1906.02717,* 2019.

[8] K. Zhang, Z. Yang, H. Liu, T. Zhang and T. Başar, "Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents," *arXiv:1802.08757v2,* 2018.

[9] Y. Hu, M. Chen, W. Saad, H. V. Poor and S. Cui, "Distributed Multi-agent Meta Learning for Trajectory Design in Wireless Drone Networks," *arXiv:2012.03158,* 2020.

[10] S. Sun and M. Kiran, "Multi-Agent Meta Reinforcement Learning for Packet Routing in Dynamic Network Environments," in *The International Conference for High Performance Computing, Networking, Storage, and Analysis*, 2020.

[11] M. Botvinick, S. Ritter, J. X. Wang, Z. Kurth-Nelson, C. Blundell and D. Hassabis, "Reinforcement Learning, Fast and Slow," *Trends in Cognitive Sciences,* vol. 23, no. 5, pp. 408-422, 2019.

[12] C. Zhang, O. Vinyals, R. Munos and S. Bengio, "A Study on Overfitting in Deep Reinforcement Learning," *arXiv:1804.06893,* 2018.

[13] A. Raghu, M. Raghu, S. Bengio and O. Vinyals, "Rapid Learning or Feature Reuse? Towards Understanding the Effectiveness of MAML," *International Conference of Learning Representations,* 2020.

[14] T. Salimans, J. Ho, X. Chen and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," *arXiv:1703.03864,* 2017.

[15] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. D'Oliveira, H. Eichner, S. E. Rouayheb, D. Evans and J. Gardner, "Advances and Open Problems in Federated Learning," *arXiv:1912.04977,* 2019.

[16] Jeromemetronome, "Wikipedia," 6 June 2019. [Online]. Available: https://en.wikipedia.org/wiki/File:Federated_learning_process_central_case.png. [Accessed 7 May 2021].