PEER-TO-PEER MODEL-AGNOSTIC META-LEARNING

Muhammad I. Qureshi and Usman A. Khan

Tufts University, Medford, MA

2024 IEEE 13th Sensor Array and Multichannel Signal Processing Workshop (SAM) | 979-8-3503-4481-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/SAM60225.2024.10636527

ABSTRACT

In this paper, we propose two distributed meta-learning methods implementable over a distributed peer-to-peer network. We consider a distributed problem where each computational node possesses a local model along with a private dataset used for training that model. Every node aims to minimize the local loss using the private data and the global loss through a weight-mixing strategy. All nodes undergo training using a few-shot multi-task learning method: model-agnostic meta-learning (MAML). We consider *adapt, learn, and share* methodology, where each model *adapts* to some private data samples to capture generalizable characteristics. Subsequently, each model *learns* by minimizing the loss using a different set of samples from the private dataset. Finally, each node *shares* certain model parameters with the neighboring nodes and updates the local parameters through aggregation. The goal is to train the local models across a diverse range of tasks, enabling them to quickly learn new tasks using a limited set of training examples. We use node-level MAML and network-level weight-mixing techniques for few-shot multi-task distributed meta-learning. We show numerical experiments to illustrate the performance of the proposed methods on real-world datasets.

***Index Terms*—** Distributed optimization, model-agnostic meta-learning, few-shot learning, peer-to-peer networks.

1. INTRODUCTION

Machine learning methods gained huge interest in the past decade due to their applications in signal processing, image classification, and robotics [1–4]. The accuracy of conventional methods depends upon the access to information. For optimal performance, they require large-scale datasets to train the models. This is usually not feasible because, in many practical applications, we either do not have enough data to train the model or the data is geographically distributed. To solve the first problem, several few-shot multi-task learning methods were proposed [5–7]. The most significant recent work discusses a model agnostic meta-learning (MAML) [7] framework, which leverages gradient descent to understand the commonalities among various tasks such that a few examples from a new task will produce good results. The second problem (of data distribution) can be solved using various weight-mixing methodologies [8, 9].

Conventional machine learning methods are trained on a dataset to learn a single task (e.g., the classification of images or text). Although the performance of such models drastically improved over the years [2,10], they cannot adapt to new tasks with limited data examples. To this aim, several multi-task learning methods were introduced in [5,6] for facilitating rapid task adaptation. This involves training a model on multiple related or unrelated tasks (e.g., facial expression and age estimation using a single model). Some methods achieve this goal by sharing features learned from different tasks, while some models are trained using a joint loss function (the model minimizes the combined loss for multiple tasks).

Meta-learning [11, 12] proposes another approach to improve the model's capabilities to generalize well. The goal is to acquire generalizable knowledge from a diverse set of tasks and then use it to learn new (possibly unseen) tasks from a few examples. The model generally adapts its parameters using a few data samples (known as the support set) and then it utilizes these adapted parameters and another set of data samples (often called the query set) to update the model parameters. This methodology is also known as "*learning to learn*".

MAML [13–16] combines both multi-task learning and meta-learning while using a diverse set of tasks for training the model. It learns a good initialization (using gradient descent steps) and quickly adapts to new tasks using a few data samples. While MAML has been extensively studied for its flexibility and efficiency, it faces limitations when dealing with geographically distributed data. In such scenarios, we need new strategies to efficiently train the models.

Distributed methods are well-studied in the literature on optimization theory [8, 9] due to their diverse applications in signal processing, controls, and game theory [17–20]. Several distributed architectures (based on network connectivity) have been explored. For well-connected environments, federated learning methods [8, 21] gained popularity. These methods rely upon several client devices connected to a central server. Although useful, this architecture is vulnerable due to a single point of failure (the central server). Some work on arbitrary peer-to-peer (P2P) networks can be found in [22, 23].

The authors are with the ECE Department at Tufts University; muhammad.qureshi@tufts.edu and khan@ece.tufts.edu. This work has been partially supported by NSF under award PIRE-2230630. Usman A. Khan holds concurrent appointments as a Professor at Tufts University and as an Amazon Scholar with Amazon Robotics. This paper describes work performed at Tufts University and is not associated with Amazon.

In this paper, we propose two distributed MAML methods designed for P2P networks. These methods follow the strategy to *adapt, learn, and share*. They use two weight-mixing methodologies: (i) We propose P2P−MAML, where each node shares its model parameters with the neighboring nodes; (ii) We also propose P2P−MAML+, where each node shares the model parameters and the gradients computed for loss minimization with its neighboring nodes. We show numerical experiments to illustrate that P2P−MAML outperforms independently trained local MAML by a significant margin. Moreover, we demonstrate that P2P−MAML+ surpasses the performance results of P2P−MAML using an additional gradient estimation step for updating the local model.

2. PROBLEM FORMULATION

We begin by describing the centralized model agnostic meta-learning framework, followed by an explanation of the distributed P2P problem setup.

2.1. Model Agnostic Meta Learning Framework

Meta-learning methods generally follow two steps: (i) an inner loop that evaluates meta-data; (ii) an outer loop that updates model parameters. Unlike conventional machine learning, where the target is to train the model by minimizing the loss function $\mathcal{L}$ for different batches in the training set, MAML is trained over a set of tasks $\mathcal{T} = \{\tau_1, \cdots, \tau_m\}$ following a probability distribution $p(\mathcal{T})$. We consider a parameterized function $f(\mathbf{x})$ (model) with parameters $\mathbf{x} \in \mathbb{R}^p$. Then for each task, MAML evaluates the auxiliary parameters $\widetilde{\mathbf{x}}_j \in \mathbb{R}^p$ using a single or multiple steps of gradient descent:

$$\widetilde{\mathbf{x}}_j = \mathbf{x} - \alpha \nabla \mathcal{L}_{\mathcal{T}_j}(f(\mathbf{x})), \tag{1}$$

where α is a hyperparameter that can be adaptive or fixed. This task-based initialization ($\widetilde{\mathbf{x}}_j$) is then used to learn

$$\min_{\mathbf{x}} \left\{ \textstyle\sum_{\tau_j \sim p(\mathcal{T})} \mathcal{L}_{\tau_j}(f(\widetilde{\mathbf{x}}_j)) = \sum_{j=1}^{m} p(\tau_j) \mathcal{L}_j(f(\widetilde{\mathbf{x}}_j)) \right\}.$$

Thus, for learning rate β, meta-optimization is performed across all tasks to evaluate new model parameters

$$\mathbf{x} \leftarrow \mathbf{x} - \beta \cdot \textstyle\sum_{\tau_j \sim p(\mathcal{T})} \nabla \mathcal{L}_{\tau_j}(f(\widetilde{\mathbf{x}}_j)). \tag{2}$$

As shown in Figure 1, each task τ_j is split into a *support set* $\mathcal{D}_j^S$ and a *query set* $\mathcal{D}_j^Q$, where $j \in \{1, \cdots, m\}$. The support set helps the model in acquiring a good initialization $\widetilde{\mathbf{x}}_j$. Subsequently, the model parameters are updated by minimizing the task-specific loss functions, utilizing the new initialization ($\widetilde{\mathbf{x}}_j$) along with the data samples from the query set. The model is updated using *adapt and learn* strategy:

- *Adapt:* Evaluate auxiliary parameters

$$\widetilde{\mathbf{x}}_j = \mathbf{x} - \alpha \nabla \mathcal{L}_{\tau_j}(f(\mathbf{x}), \mathcal{D}_j^S), \tag{3}$$

- *Learn:* Update model parameters

$$\mathbf{x} \leftarrow \mathbf{x} - \beta \cdot \textstyle\sum_{\tau_j \sim p(\mathcal{T})} \nabla \mathcal{L}_{\tau_j}(f(\widetilde{\mathbf{x}}_j), \mathcal{D}_j^Q). \tag{4}$$

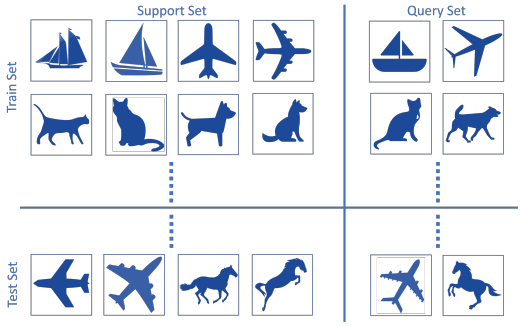

Fig. 1. MAML data segmentation.

The model adapts the initial parameters using the support set (3) and updates the (initialized) model parameters using the query set (4). This learning methodology helps the model to comprehend new tasks very quickly.

2.2. Distributed Learning Setup

The distributed learning methods consider a global problem divided among n nodes communicating over a strongly connected network. The global problem aims to minimize the average of local loss functions and can be written as:

$$\mathbf{P}: \quad \min_{\mathbf{x} \in \mathbb{R}^p} \left\{ \mathcal{L}(\mathbf{x}) := \tfrac{1}{n} \textstyle\sum_{i=1}^{n} \mathcal{L}_i(\mathbf{x}) \right\},$$

where $\mathcal{L}_i$ is local loss function private to node i such that $i \in \{1, \cdots, n\}$. Several distributed optimization methods are proposed [9,24] to solve $\mathbf{P}$. Of significance is distributed gradient descent (DGD) which uses a weight aggregation strategy for model parameters based on network connectivity. We define $W = \{w_{i,j}\}$ as the weight matrix that represents the network connectivity and let $\mathbf{x}_i^k$ be the state vector estimate of model parameters at node i evaluated at k-th iteration. Then at each iteration, DGD updates as follows:

$$\mathbf{x}_i^k = \textstyle\sum_{r=1}^{n} w_{i,r} \left(\mathbf{x}_r^{k-1} - \alpha \nabla \mathcal{L}_i(\mathbf{x}_r^{k-1}) \right), \qquad \forall k > 1.$$

This weight-mixing strategy is widely recognized in the literature on distributed optimization for its straightforward implementation and notable enhancement of performance results. However, it can be theoretically shown [17] that the local models do not converge to the solution of the global problem $\mathbf{P}$ due to the heterogeneous nature of data distribution among the network of nodes. To fix this error, [9] discusses a gradient tracking technique to estimate the global gradient iteratively. At each node i, the proposed method evaluates:

$$\mathbf{y}_i^{k+1} = \textstyle\sum_{r=1}^{n} w_{i,r} \left(\mathbf{y}_r^k + \nabla \mathcal{L}_i(\mathbf{x}_r^{k+1}) - \nabla \mathcal{L}_i(\mathbf{x}_r^k) \right). \tag{5}$$

This results in updating the local state estimates $\mathbf{x}_i^k$ to move in the negative of the direction of gradient tracking term $\mathbf{y}_i^k$. Furthermore, it can be verified [9] that at each node, the gradient tracking term converges to the gradient of the global problem, i.e., $\mathbf{y}_i^k \to \nabla \mathcal{L}$.

2.3. Distributed Model Agnostic Meta Learning

In this section, we propose a distributed MAML framework that uses the advantages of multi-task meta-learning as well as the flexibility of distributed setups. We consider a P2P network of n nodes communicating over a strongly connected graph. Each node i possesses a local dataset $\mathcal{D}_i = \{\mathcal{D}_i^S, \mathcal{D}_i^Q\}$ consisting of tasks $\mathcal{T}_i = \{\tau_{i,1}, \cdots, \tau_{i,m_i}\}$ (provided that each node i possesses m_i tasks) sampled following the probability distribution $p(\mathcal{T}_i)$. We denote $\mathcal{D}_{i,j} = \{\mathcal{D}_{i,j}^S, \mathcal{D}_{i,j}^Q\}$ as the set of data samples used for training the local model $f_i(\mathbf{x})$ on the j-th task. Each node aims to minimize its loss by adaptation using the support set and then learning new parameters using the query set as described below:

$$\widetilde{\mathbf{x}}_{i,j} = \mathbf{x}_i - \alpha \nabla \mathcal{L}_{\tau_{i,j}}(f_i(\mathbf{x}_i), \mathcal{D}_{i,j}^S), \tag{6}$$

$$\mathbf{x}_i \leftarrow \mathbf{x}_i - \beta \cdot \nabla \left(\textstyle\sum_{\tau_{i,j} \sim p(\mathcal{T}_i)} \mathcal{L}_{\tau_{i,j}}(f(\widetilde{\mathbf{x}}_{i,j}), \mathcal{D}_{i,j}^Q) \right), \tag{7}$$

for all $i \in \{1, \cdots, n\}$, and $j \in \{1, \cdots, m_i\}$. The distributed MAML problem can be mathematically written as:

$$\mathbf{P}: \quad \min_{\mathbf{x} \in \mathbb{R}^p} \{\mathcal{L}(\mathbf{x}) := \tfrac{1}{n} \textstyle\sum_{i=1}^n \mathcal{L}_i(\mathbf{x})\},$$
$$\text{where} \quad \mathcal{L}_i(\mathbf{x}) := \textstyle\sum_{j=1}^{m_i} p(\tau_{i,j}) \cdot \mathcal{L}_{\tau_{i,j}}(f_i(\mathbf{x}), \mathcal{D}_{i,j}).$$

2.4. P2P-MAML and P2P-MAML+

In this paper, we propose two distributed model-agnostic meta-learning methods that use weight-mixing of model parameters (P2P-MAML) and gradient tracking (P2P-MAML+). Firstly, we describe P2P-MAML, such that each node undertakes the meta-initialization process by *adapting* to the support set as described in (6). Subsequently, it *learns* new model parameters using query set, as outlined in (7). Finally, every node *shares* its model parameters with the neighboring nodes and evaluates a weighted aggregation based on the network connectivity. In addition to P2P-MAML, P2P-MAML+ evaluates a gradient tracking term $\mathbf{y}_i$ (at each node) similar to (5). This gradient estimate is shared with neighboring nodes, and an aggregation of gradients is then used to evaluate gradient descent, leading to the update of the local parameters.

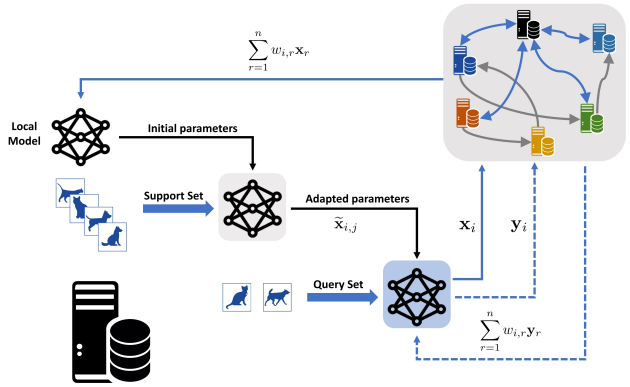

Fig. 2. Node level update procedure for peer-to-peer distributed model-agnostic meta-learning P2P-MAML+.

Figure 2 describes an update of P2P-MAML+ where the black arrows symbolize node-level updates, while the blue arrows represent network-level interactions. The dashed lines depict the steps involved in estimating the global gradient. Omitting these steps would lead to the representation of P2P-MAML.

Algorithm 1 P2P-MAML at each node i

Require: $\mathbf{x}_i^0 \in \mathbb{R}^p, \alpha, \beta > 0, \{w_{ij}\}, \mathcal{D}_i, p(\mathcal{T}_i)$

for $k = 0, 1, 2, \cdots$ **do**
  Sample batch of tasks $\tau_{i,j} \sim p(\mathcal{T}_i)$
  for all $\tau_{i,j}$ **do**
    $\widetilde{\mathbf{x}}_{i,j} \leftarrow \mathbf{x}_i^k - \alpha \cdot \nabla \mathcal{L}_{\tau_{i,j}}(f_i(\mathbf{x}_i^k, \mathcal{D}_{i,j}^S))$
  end for
  $\mathbf{x}_i^{k+1} \leftarrow \sum_{r=1}^n w_{ir} \big(\mathbf{x}_r^k - \beta \cdot \nabla \sum_{\tau_{i,j} \sim \mathcal{T}_i} \mathcal{L}_{\tau_j}(f_i(\widetilde{\mathbf{x}}_{i,j}, \mathcal{D}_{i,j}^Q))\big)$
end for

P2P-MAML is formally described in Algorithm 1. We assume that the weight matrix representing the network connectivity $W = \{w_{ir}\}$ is doubly stochastic (a common assumption in the literature on distributed optimization [18]). Each node initializes the local model parameters $\mathbf{x}_i$ randomly. In each iteration, every node calculates task-specific meta-learning parameters $\widetilde{\mathbf{x}}_{i,j}$ through a gradient descent step (or multiple steps). These parameters are computed using the support set associated with each task $\tau_{i,j}$, where the tasks are sampled from the node's local task set $\mathcal{T}_i$ with a probability of $p(\mathcal{T}_i)$. Subsequently, the model parameters are updated using the query set $\mathcal{D}_{i,j}^Q$ and $\widetilde{\mathbf{x}}_{i,j}$. Finally, each node shares its local $\mathbf{x}_i$ and collectively updates them through a weighted aggregation process. This weight-mixing strategy encourages each node to acquire knowledge from the neighboring nodes, leading to a significant improvement in the performance of local models when faced with new (unseen) tasks.

Algorithm 2 P2P-MAML+ at each node i

Require: $\mathbf{x}_i^0 \in \mathbb{R}^p, \alpha, \beta > 0, \{w_{ij}\}, \mathcal{D}_i, p(\mathcal{T}_i)$

for $k = 1, 2, 3, \cdots$ **do**
  Sample batch of tasks $\tau_{i,j} \sim p(\mathcal{T}_i)$
  for all $\tau_{i,j}$ **do**
    $\widetilde{\mathbf{x}}_{i,j}^k \leftarrow \mathbf{x}_i^k - \alpha \cdot \nabla \mathcal{L}_{\tau_{i,j}}(f_i(\mathbf{x}_i^k, \mathcal{D}_{i,j}^S))$
  end for
  $\mathbf{t}_i^k \leftarrow \nabla \sum_{\tau_{i,j} \sim \mathcal{T}_i} \mathcal{L}_{\tau_{i,j}}(f_i(\widetilde{\mathbf{x}}_{i,j}^k, \mathcal{D}_{i,j}^Q))$
  If $k = 1$: $\mathbf{y}_i^k \leftarrow \mathbf{t}_i^k$
  Else: $\mathbf{y}_i^k \leftarrow \sum_{r=1}^n w_{ir} \left(\mathbf{y}_r^{k-1} + \mathbf{t}_r^k - \mathbf{t}_r^{k-1}\right)$
  $\mathbf{x}_i^{k+1} \leftarrow \sum_{r=1}^n w_{ir} \left(\mathbf{x}_r^k - \beta \cdot \mathbf{y}_r^k\right)$
end for

Algorithm 2 formally describes P2P-MAML+ method. The first iteration is similar to Algorithm 1. Starting from the second iteration onwards, it uses an additional gradient tracking step, where $\mathbf{y}_i^k$ estimates the gradient of the global loss $\mathcal{L}$ using the current $\mathbf{t}_i^k$ and the previous local gradients $\mathbf{t}_i^{k-1}$. This gradient estimate $\mathbf{y}_i^k$ is then used by each node to update its local models.

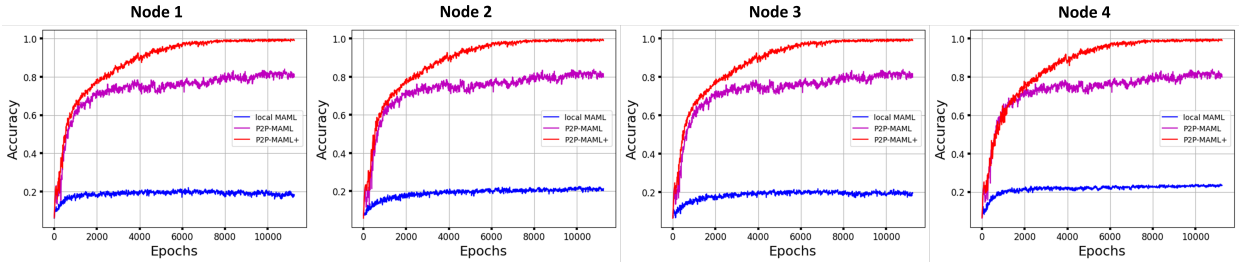

Fig. 3. Performance comparison of local MAML with P2P−MAML and P2P−MAML+. Each plot shows the classification accuracy of each node for the Omniglot dataset using 20-way 5-shots setup.

Datasets	Methods	8-way 5-shots	8-way 1-shot	20-way 5-shots	20-way 1-shot
Omniglot [25]	local MAML	24.3 %	24.1 %	23.2 %	21.9 %
	P2P−MAML	91.9 %	87.3 %	85.7 %	82.5 %
	P2P−MAML+	**99.9 %**	**99.8 %**	**98.9 %**	**98.5 %**
CIFAR-FS [26]	local MAML	23.5 %	22.8 %	18.4 %	12.7 %
	P2P−MAML	84.7 %	81.4 %	72.6 %	58.5 %
	P2P−MAML+	**88.8 %**	**86.7 %**	**79.3 %**	**68.2 %**

Table 1. Average accuracy for classification of Omniglot and CIFAR-FS datasets over all nodes.

3. EXPERIMENTAL EVALUATION

In this section, we illustrate the performance of proposed methods using different numerical experiments. The goal is to evaluate the improvement in accuracy of local models using P2P−MAML and P2P−MAML+ as compared to the centralized implementation. For consistency with prior work, we follow the standards for few-shot learning benchmarks and consider different permutations of *N-way K-shots* training, i.e., each task has N classes and K instances of each class.

In our problem setup, we consider a network of four nodes communicating over a P2P network. Each node can communicate with its neighbors but sharing of training data is prohibited. Furthermore, we assume that the local data has non-overlapping tasks (the i-th node has no knowledge of the tasks possessed by the other nodes). The goal is to train the local models using private dataset and a weight-mixing strategy, such that they accurately classify the tasks possessed by all nodes. We next describe the architecture of local models.

To ensure fair evaluations, we use an identical model architecture for each node, which consists of 4 Convolutional modules connected serially as discussed in [27]. Each module has: (i) a convolution layer and (ii) batch normalization followed by (iii) a rectified linear unit and (iv) max pooling. The dimensionality of each hidden layer is set to 64 and the hyperparameters are $\alpha = \beta = 0.01$. We focus on image classification tasks, where the number of channels in the Convolutional module is chosen based on the type of images.

3.1. Classification results

We consider the classification accuracy of images from Omniglot [25] and CIFAR-FS [26] datasets. Omniglot contains 1623 handwritten characters drawn by 20 people with a total of 50 unique alphabets, while CIFAR-FS has images from 100 classes, selected from the CIFAR-100 dataset. We consider the classification problems for (i) 8-way 5-shots, (ii) 8-way 1-shot, (iii) 20-way 5-shots, and (iv) 20-way 1-shot training setups. For each of the above, every node possesses one-fourth of the total data. To model heterogeneity, we assume that the classes are unique among neighboring nodes. We only plot the results for 20-way 5-shots setup for Omniglot dataset due to space limitation.

Figure 3 illustrates the performance comparison of local MAML, P2P−MAML, and P2P−MAML+ using Omniglot dataset. The blue curve shows the accuracy for local MAML when the model parameters are not shared with the neighbors. The magenta and the red curves depict the performance of P2P−MAML and P2P−MAML+ methods. They both outperform local MAML with a significant margin, using parameter sharing and aggregation strategies. Notably, the accuracy of P2P−MAML+ is comparable to the centralized MAML [7]. However, the weight-mixing strategies add to the communication bandwidth (to an order of magnitude that is common in federated learning [8]). We further tabulate the accuracy of all aforementioned methods in Table 1 for Omniglot and CIFAR-FS datasets. P2P−MAML achieves better performance results compared to the local MAML, while P2P−MAML+ consistently outperforms the other methods.

4. CONCLUSION

In this paper, we propose two distributed MAML methods for learning over P2P networks. We use *adapt, learn, and share* methodology. Each node evaluates an initialization by adapting the model parameters using the support set. Subsequently, it learns new parameters through the minimization of the local loss, using the adapted parameters and the query set. Finally, each node shares these newly learned model parameters, which are then aggregated based on the network connectivity. We propose P2P−MAML and P2P−MAML+ methods and provide numerical experiments to illustrate their performance advantage over local MAML method.

5. REFERENCES

[1] K. O'Shea and R. Nash, "An introduction to convolutional neural networks.," *CoRR*, 2015.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conf. on Comp. Vision and Pattern Recognition*, 2016.

[3] J. Chung, N. Lawrance, and S. Sukkarieh, "Learning to soar: Resource-constrained exploration in reinforcement learning," *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 158–172, 2015.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," 2015, p. 436–444, Nature 521.

[5] R. Caruana, *Multitask Learning*, pp. 95–133, Springer US, Boston, MA, 1998.

[6] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *Int. Conf. on Learning Representations*, 2016.

[7] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. of the 34th Int. Conf. on Machine Learning - Volume 70*, 2017, ICML, p. 1126–1135.

[8] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Int. Conf. on Artificial Intelligence and Statistics*, 2016.

[9] R. Xin, S. Pu, A. Nedić, and U. A. Khan, "A general framework for decentralized optimization with first-order methods," *Proceedings of the IEEE*, vol. 108, pp. 1869–1889, 2020.

[10] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in NeurIPS*, 2012, vol. 25.

[11] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *IEEE Transactions on Pattern Analysis amp; Machine Intelligence*, vol. 44, no. 09, pp. 5149–5169, sep 2022.

[12] L. Metz, N. Maheswaranathan, B. Cheung, and J. Sohl-Dickstein, "Learning unsupervised learning rules," in *Int. Conf. on Learning Representations*, 2019.

[13] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *CoRR*, 2018.

[14] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "Meta-learning framework with applications to zero-shot time-series forecasting," in *AAAI Conference on Artificial Intelligence*, 2020.

[15] V. K. Verma, D. Brahma, and P. Rai, "Meta-learning for generalized zero-shot learning," in *AAAI Conference on Artificial Intelligence*, 2020.

[16] K. Killamsetty, C. Li, C. Zhao, R. K. Iyer, and F. Chen, "A nested bi-level optimization framework for robust few shot learning," in *AAAI Conference on Artificial Intelligence*, 2020.

[17] M. I. Qureshi, R. Xin, S. Kar, and U. A. Khan, "A decentralized variance-reduced method for stochastic optimization over directed graphs," in *2021 IEEE Int. Conf. on Acoustics, Speech and Sig. Proc. (ICASSP)*, 2021.

[18] R. Xin, S. Kar, and U. A. Khan, "Decentralized stochastic optimization and machine learning," *IEEE Signal Processing Magazine*, May 2020.

[19] M. I. Qureshi and U. A. Khan, "Distributed saddle point problems for strongly concave-convex functions," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 9, pp. 679–690, 2023.

[20] A. Beznosikov, G. Scutari, A. Rogozin, and A. Gasnikov, "Distributed saddle-point problems under data similarity," in *Advances in NeuIPS*, 2021.

[21] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," Red Hook, NY, USA, 2020, NIPS'20, Curran Associates Inc.

[22] M. Kayaalp, S. Vlaski, and A. Sayed, "Dif-MAML: Decentralized multi-agent meta-learning," *IEEE Open Journal of Signal Processing*, vol. 3, pp. 71–93, 2022.

[23] X. Zhang, C. Hu, B. He, and Z. Han, "Distributed reptile algorithm for meta-learning over multi-agent systems," *IEEE Transactions on Signal Processing*, vol. 70, pp. 5443–5456, 2022.

[24] S. S. Ram, A. Nedić, and V. V. Veeravalli, "Distributed stochastic subgradient projection algorithms for convex optimization," *Journal of Optimization Theory and Applications*, vol. 147, no. 3, pp. 516–545, 2010.

[25] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, pp. 1332–1338, 2015.

[26] L. Bertinetto, J. F. Henriques, P. Torr, and A. Vedaldi, "Meta-learning with differentiable closed-form solvers," in *Int. Conf. on Learning Representations*, 2019.

[27] T. Deleu, T. Würfl, M. Samiei, J. P. Cohen, and Y. Bengio, "Torchmeta: A Meta-Learning library for PyTorch," 2019.