2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Distributed First-Order and Second-Order Adaptive Hybrid Optimization for Multi-Robot Learning

Yilun Zhang, Xianghua Xie*, and Lu Zhang

***Abstract*— We present a distributed first-order and second-order adaptive hybrid optimization algorithm (DAHO) for multi-robot systems. A team of robots collaboratively trains a shared deep neural network using only local data while exchanging model updates via peer-to-peer robot communication. Raw data never leaves the device, which preserves privacy and conserves communication bandwidth. The method blends a second-order Limited memory Broyden–Fletcher–Goldfarb–Shann (LBFGS) method with an alternating direction method of multipliers (ADMM) based first-order method to obtain both the fast convergence of second-order methods and the robustness of first-order schemes. An automatic switching policy, guided by a convergence analysis rooted in trust region theory, selects the update type at each round. A soft switch mechanism derived from the same analysis mitigates oscillations during mode changes. Compared with four single-method baselines that range from first-order to second-order optimization, the proposed hybrid approach achieves faster convergence, superior accuracy, and near centralized performance on robotics related deep learning tasks.**

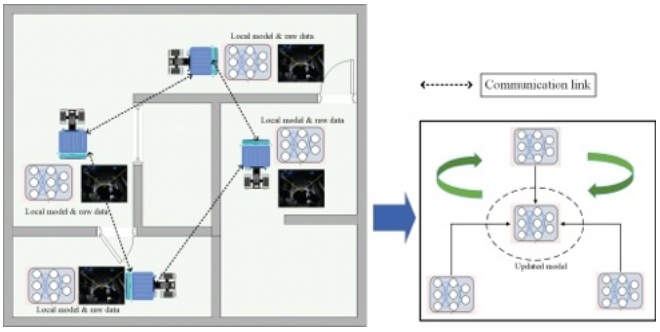


Fig. 1: Distributed learning framework: Each robot scans a segment of the environment and independently trains a neural network for neural implicit mapping of the building using its locally collected LiDAR data. Through communication links, robots exchange model updates and collaboratively optimize a shared neural network without sharing raw sensor data.

I. INTRODUCTION

Multi-robot systems enable a group of robots to collectively explore the environment and perform a common task (e.g., multi-robot implicit mapping, multi-UAV manipulation control, multi-object recognition, and cooperative autonomous driving) [1]–[5] using partially distributed data collected by each robot. This paradigm offers fast learning, wide coverage, reduced redundancy in data collection, bandwidth preservation, and privacy protection compared with traditional centralized solutions where data from robots is aggregated at a central server or a leader robot. Through a communication network graph, geographically distributed data that make centralized learning difficult can be learned by each robot in parallel. In a typical setting, consider a team of robots scanning a building and mapping the environment collectively, as shown in Fig. 1. This collaborative approach not only enlarges coverage but also improves robustness to leader node failures and reduces the difficulty of deploying a central server in dynamic and complex environments.

Distributed optimization is a significant challenge in multi-robot systems because each robot uses only its local data yet must collaboratively learn a multi-agent deep neural network model. The goal is to train the model and minimize an empirical risk via peer-to-peer robot communication so that each one can exploit from data gathered by others. Existing distributed optimization methods such as distributed stochastic gradient descent (DSGD) [6] and distributed neural network optimization (DiNNO) [7] are based on first-order methods, which update model parameters using the gradient (first-order derivative) of the objective function. These methods are generally stable and scalable for nonconvex problems. In contrast, second-order methods, which leverage curvature information via the Hessian or its approximations (second-order derivative), such as distributed quasi-Newton (DQN) [8], can achieve faster convergence. However, second-order methods can exhibit unstable performance in large-scale distributed deep neural network (DNN), as their strong convergence guarantees apply to convex objectives (e.g. logistic regression) [9]. In addition, the modified second-order method for DNN [10] incurs higher computational and memory overhead due to the need to compute the Hessian. Due to the lower resource requirements and greater robustness, first-order methods remain widely used in multi-robot systems, although they generally converge slowly than second-order methods.

In this work, we propose distributed first-order and second-order adaptive hybrid optimization (DAHO), a hybrid algorithm that combines the robustness of first-order methods with the rapid convergence of second-order methods through an adaptive switch mechanism based on convergence analysis of trust region theory [11]. We demonstrate its effectiveness on multiple distributed learning tasks through extensive simulations. Compared to baseline approaches, our

This work is supported by an EPSRC grant with the funder reference EP/Y007697/1.

Yilun Zhang and Xianghua Xie (* corresponding author) are with the School of Mathematics and Computer Science, Swansea University, Swansea, U.K. (e-mail: yilun.zhang@swansea.ac.uk, x.xie@swansea.ac.uk).

Lu Zhang is with the Institute for Digital Technologies, Loughborough University London, London, U.K. (e-mail: l.zhang12@lboro.ac.uk).

method achieves faster convergence with fewer communication rounds and attains higher final accuracy, due to the hybrid search direction incorporated in the optimization. Unlike the warm up and switch scheme used in centralized training, which relies on a preset iteration-based switch [12], our adaptive mechanism requires no manual parameter tuning in a distributed framework. It also mitigates oscillations by replacing hard switch with a soft switch strategy derived from the convergence analysis of the automatic switch strategy. In addition, DAHO is easy to integrate with standard PyTorch tools and common optimizers such as Adam [13] and LBFGS [14].

To improve convergence in distributed optimization for multi-robot learning, our contributions are summarized as follows:

- We propose a hybrid distributed optimization method for multi-robot systems that leverages second-order method for fast convergence, while maintaining computational efficiency and stability through first-order updates in nonconvex, large-scale DNN settings.
- We develop an adaptive soft switch algorithm based on the convergence performance, providing an automatic transition from a fast converging second-order approach to a stable and robust first-order approach.
- We demonstrate the superior performance of the DAHO algorithm on a variety of distributed tasks in multi-robot scenarios, including distributed recognition, distributed implicit mapping, and distributed multi-agent reinforcement learning.

The remainder of the paper is organized as follows. Section II reviews related work in distributed learning. Section III formulates the problem. Section IV presents the proposed adaptive hybrid optimization algorithm, DAHO. Section V reports experimental results on three robotics-related DNN tasks. Section VI concludes the paper.

II. RELATED WORKS

Stochastic optimization algorithms have been extensively studied in machine learning. Classical stochastic gradient descent (SGD) [15] and its variants are widely adopted because they rely solely on gradient information to determine search directions and are effective in large-scale DNN problems. In parallel, second-order quasi Newton methods have been investigated for their superior convergence properties relative to SGD [16].

In distributed learning, both first order and second order approaches have been explored. Distributed subgradient methods have been extended to DSGD, where each agent combines local gradients with neighbors' parameters, and to distributed stochastic gradient tracking (DSGT) [17], which improves convergence by tracking the global gradient estimate. The alternating direction method of multipliers (ADMM) [18] is another popular framework for distributed optimization. Building on ADMM, DiNNO has been proposed specifically for multi robot systems, where approximate primal and dual variables are updated and exchanged among neighboring robots, yielding improved performance over DSGD and DSGT. To exploit curvature information in large scale settings, limited memory Broyden–Fletcher–Goldfarb–Shanno (LBFGS) [19] approximates the Hessian matrix and thus reduces computation and memory requirements. Distributed BFGS [20] types methods employ LBFGS updates at each node and exchange parameters to reach consensus across the network. [10] proposes a practical quasi-Newton approach for training DNN, however the method still incurs substantial memory and computational overhead.

Representative applications include multi-robot neural implicit mapping based on NeRF [21], where RGB images are used to train neural networks for environment perception tasks. Distributed frameworks have been proposed in which each robot collects range data in real time to build a local map that is then shared and integrated via wireless communication using ADMM based optimization in [22]. Real time asynchronous multi agent neural implicit mapping has also been studied under the DiNNO framework [23]. Beyond mapping, distributed optimization has been applied to multi robot control and reinforcement learning [24], for example in real time collision avoidance using offline trained value functions.

III. PROBLEM FORMULATION

We consider a DNN problem setting with N robots where each robot has access to its local problem data D_n, and the robots are connected through a communication graph $G = (\mathcal{N}, \mathcal{E})$. Robot $n \in \mathcal{N}$ can communicate with its neighbors $M_n = \{m \in \mathcal{N} \mid (n,m) \in \mathcal{E}\}$. Considering $y = f(x;\boldsymbol{\theta})$ to be the function of a DNN, where x, y, and $\boldsymbol{\theta}$ represent for the input, output, and d demension of parameters, respectively. Let $l(\cdot)$ be the respective loss function, then the distributed learning optimization can be formulated as follow:

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \sum_{n\in\mathcal{N}} l(\boldsymbol{\theta}; D_n). \tag{1}$$

Since only connected robots can exchange information through communication, the above minimization problem can be reformulated as a constrained minimization problem as follows:

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \quad \sum_{n\in\mathcal{N}} l(\boldsymbol{\theta}_n; D_n) \tag{2a}$$

$$\text{s.t.} \quad \boldsymbol{\theta}_n = \boldsymbol{\theta}_m, \ \ \forall\,(n,m) \in \mathcal{E}, \tag{2b}$$

where (2b) indicates that neighbour robots are connected and enforce agreement on the model parameters to minimize the local loss functions. For generality, the objective functions in (2a) can denote any standard DNN loss function. In the distributed consensus formulation where agreement can only be enforced between connected neighbours, (2) can be written as:

$$\begin{aligned} \min_{\{\boldsymbol{\theta}_n\},\{\mathbf{z}_{nm}\}} \quad & \sum_{n\in\mathcal{N}} l(\boldsymbol{\theta}_n, D_n) \\ \text{s.t.} \quad & \boldsymbol{\theta}_n = \mathbf{z}_{nm}, \ \ \boldsymbol{\theta}_m = \mathbf{z}_{nm}, \ \ \forall\,(n,m) \in \mathcal{E}, \end{aligned} \tag{3}$$

where z_{nm} is a auxiliary variable to guarantee consensus on neighboring robots n and m.

IV. DISTRIBUTED OPTIMIZATION ANALYSIS

To update each robot's DNN parameters so they can collectively learn a shared model from distributed data, the distributed optimization problem should be analyzed. In this section, we begin by outlining the theoretical background of DAHO, followed by a detailed description of the proposed optimization algorithm and baseline algorithms.

A. ADMM Based First-Order Optimization Method

ADMM is a popular method for solving distributed optimization problems. It operates by iteratively updating the primal and dual variables while exchanging them with neighboring agents (robots) via communication. The augmented Lagrangian for (3) can be written as:

$$\mathcal{L}_a = \sum_{n\in\mathcal{N}} \Big(l(\boldsymbol{\theta}_n) + \sum_{m\in M_n} (\boldsymbol{\lambda}_{nm}^n)^\top (\boldsymbol{\theta}_n - \mathbf{z}_{nm}) + (\boldsymbol{\lambda}_{nm}^m)^\top (\boldsymbol{\theta}_m - \mathbf{z}_{nm}) + \frac{\rho}{2}\|\boldsymbol{\theta}_n - \mathbf{z}_{nm}\|_2^2 + \frac{\rho}{2}\|\boldsymbol{\theta}_m - \mathbf{z}_{nm}\|_2^2 \Big), \tag{4}$$

where $\lambda_{nm}^n, \lambda_{nm}^m$ are the dual variables, ρ is a positive penalty parameter is the step size of gradient ascent of the dual variable to make consensus. Define $\boldsymbol{p}_n = \sum_{m\in M_n} \lambda_{nm}^n + \lambda_{nm}^m$ which combines all dual variables associated with $\boldsymbol{\theta}_n$. Let the auxiliary variable $\mathbf{z}_{nm} = \frac{1}{2}(\boldsymbol{\theta}_n + \boldsymbol{\theta}_m)$ and t represent the iteration step. The updates of ADMM can be formulated as follows:

$$\boldsymbol{\theta}_n^{t+1} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, l(\boldsymbol{\theta}; D_n) + \boldsymbol{\theta}_n^\top \boldsymbol{p}_n^t + \rho \sum_{m\in M_n} ||\boldsymbol{\theta}_n - \frac{\boldsymbol{\theta}_n^t + \boldsymbol{\theta}_m^t}{2}||_2^2, \tag{5}$$

$$\boldsymbol{p}_n^{t+1} = \boldsymbol{p}_n^t + \rho \sum_{m\in M_n} (\boldsymbol{\theta}_n^{t+1} - \boldsymbol{\theta}_m^{t+1}), \tag{6}$$

where (5) is refered as the primal update and (6) is refered as the dual update.

B. BFGS Based Second-Order Optimization Method

Assume f is continuously differentiable. Let x_t denote the current iterate and $g_t = \nabla f(x_t)$ its gradient. The BFGS [25] method maintains $\boldsymbol{H}_t \approx \nabla^2 f(x_t)^{-1}$ and chooses the search direction $p_t = -\boldsymbol{H}_t g_t$ with step size $\alpha_t > 0$:

$$x_{t+1} = x_t + \alpha_t p_t. \tag{7}$$

Define the difference of output and gradient vectors $s_t = x_{t+1} - x_t,\ y_t = g_{t+1} - g_t$. BFGS methods accelerate first–order optimization by constructing a symmetric, positive–definite approximation to the inverse Hessian $\boldsymbol{H}_t$ constructed from the curvature pairs (s_t, y_t) that satisfies the *secant equation*: $\boldsymbol{H}_{t+1} y_t = s_t$.

The Hessian $\boldsymbol{H}_t$ approximation $\boldsymbol{B}_t \approx \nabla^2 f(x_t)$ can be written as:

$$\boldsymbol{B}_{t+1} = \boldsymbol{B}_t - \frac{\boldsymbol{B}_t s_t s_t^\top \boldsymbol{B}_t}{s_t^\top \boldsymbol{B}_t s_t} + \frac{y_t y_t^\top}{y_t^\top s_t}. \tag{8}$$

For large–scale problems, forming and storing $\boldsymbol{H}_t$ (or $\boldsymbol{B}_t$) is computationally prohibitive. LBFGS circumvents this by keeping only the most recent m curvature pairs $\{(s_i, y_i)\}_{i=t-m}^{t-1}$ and computing $p_t = -\boldsymbol{H}_t g_t$ through the standard two–loop recursion without explicitly forming $\boldsymbol{H}_t$. The recursion requires $\mathcal{O}(md)$ storage and time for d-dimensional parameters, while retaining much of the fast convergence of second–order methods and the modest memory footprint of first–order algorithms. The iteration update follows (8).

C. Distributed First-Order and Second-Order Adaptive Hybrid Optimization (DAHO)

In nonconvex, large-scale DNN problems, although second-order methods can achieve rapid initial convergence, the performance of LBFGS is still inferior compared to more reliable first-order methods. Moreover, the computation and memory required for evaluating the Hessian at every iteration can be substantial. To leverage the advantages of both approaches while reducing computational complexity, we propose a hybrid optimization algorithm that automatically switches from an LBFGS based method to an ADMM based method. The switch criterion is determined adaptively by predicting gradient improvement inspired by the convergence analysis of the LBFGS trust-region method [11], rather than relying on a preset rule such as a fixed number of iterations [12]. Furthermore, we introduce a modified LBFGS that aligns more closely with the ADMM based algorithm, thereby reducing oscillations during the transition. A soft switch strategy, also derived from the trust-region analysis, is designed to further smooth the switching process.

a) Distributed ADMM based method (DADMM): Our DADMM method uses a similar update of the dual variables as described in Section A. To perform the update in (5), the primal step is approximated following the idea of DiNNO [7], where the primal update is obtained through K iterations indexed by $k = 0, \ldots, K-1$:

$$\begin{aligned} \boldsymbol{\Theta}_n^{k+1} &= \boldsymbol{\Theta}_n^k + G\Big(\boldsymbol{\Theta}_n^k;\, \rho,\, p_n^{t+1},\, \boldsymbol{\Theta}_n^t,\, \{\boldsymbol{\Theta}_m^t\}_{m\in M_n},\, D_n\Big), \\ \boldsymbol{\theta}_n^{t+1} &= \boldsymbol{\Theta}^K, \end{aligned} \tag{9}$$

where G represents the stochastic gradient over D_n.

b) Dual variables modified LBFGS (MLBFGS): To ensure a smooth transition from the second-order method to our first-order method in a), rather than using the weighting strategy [20], we adopt the LBFGS method augmented with the dual variables from ADMM to promote agreement among neighbours. Specifically, we introduce dual variables $\boldsymbol{\Lambda}_n$ for each node, and at iteration t these variables are updated by gradient ascent:

$$\boldsymbol{\Lambda}_n^{t+1} \;=\; \boldsymbol{\Lambda}_n^t \;+\; \rho \sum_{m\in M_n} \big(\boldsymbol{\theta}_n - \boldsymbol{\theta}_m\big), \tag{10}$$

where $\rho > 0$ is the ADMM step size.

Given $\boldsymbol{\Lambda}_n^t$, each node solves an augmented local subproblem that implicitly enforces consensus without requiring explicit neighbor averaging. The local augmented Lagrangian is given by:

$$l_n^t(\boldsymbol{\theta}_n) \;=\; l_n(\boldsymbol{\theta}_n) \;+\; \boldsymbol{\Lambda}_n^{\top\, t}\boldsymbol{\theta}_n \;+\; \rho \Big\| \sum_{m\in M_n} \big(\boldsymbol{\theta}_n - \boldsymbol{\theta}_m\big)\Big\|_2^2, \tag{11}$$

whose gradient with respect to $\boldsymbol{\theta}_n$ is:

$$\nabla_{\boldsymbol{\theta}_n} l_n^t(\boldsymbol{\theta}_n) \;=\; \nabla l_n(\boldsymbol{\theta}_n) \;+\; \boldsymbol{\Lambda}_n^t \;+\; 2\rho \sum_{m\in M_n} \big(\boldsymbol{\theta}_n - \boldsymbol{\theta}_m\big). \tag{12}$$

Rather than taking a fixed primal step in ADMM, we apply a few quasi-Newton iterations of LBFGS to $l_n^t(\boldsymbol{\theta}_n)$, thereby constructing a limited-memory inverse Hessian approximation $\boldsymbol{H}_n^t \approx \big(\nabla^2 l_n^t\big)^{-1}$. Therefore, the fast convergence of the second-order method is leveraged together with a strong consensus penalty adapted from ADMM. In addition, the introduced dual variable matches the formulation in a), making the switch between phases smoother and less abrupt. A single LBFGS update at node n is given by:

$$\boldsymbol{\theta}_n^{t+1} \;=\; \boldsymbol{\theta}_n^t \;-\; \boldsymbol{H}_n^t\, \nabla_{\boldsymbol{\theta}_n} l_n^t\big(\boldsymbol{\theta}_n^t\big), \tag{13}$$

and can be repeated for a small, fixed number of inner iterations (e.g., 5–10) to approximately minimize the augmented subproblem.

c) Switch criteria and soft switch: To decide when to switch from the MLBFGS in b) to the robust DADMM in a), we measure the agreement between the *predicted* and *actual* reduction of the loss function.

Let $f(\boldsymbol{\theta})$ denote the augmented loss. At iteration t, MLBFGS maintains a local quadratic model [26] $m_t = f(\boldsymbol{\theta}_t) + \boldsymbol{g}_t^\top \boldsymbol{s}_t + \frac{1}{2}\,\boldsymbol{s}_t^\top \boldsymbol{B}_t \boldsymbol{s}_t$. Defining the step and curvature estimates $\boldsymbol{s}_t = \boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t, \boldsymbol{y}_t = \nabla f(\boldsymbol{\theta}_{t+1}) - \nabla f(\boldsymbol{\theta}_t)$, and approximating the Hessian $\boldsymbol{B}_t$ as gradient differences $y_t^\top s_t$, the predicted reduction is:

$$\mathrm{pred}_t \;=\; f(\boldsymbol{\theta}_t) - m_t(\boldsymbol{s}_t) \;\approx\; -\boldsymbol{g}_t^\top \boldsymbol{s}_t \;-\; \tfrac{1}{2}\,\boldsymbol{s}_t^\top \boldsymbol{y}_t, \tag{14}$$

where $\boldsymbol{g}_t = \nabla f(\boldsymbol{\theta}_t)$. The actual reduction is measured directly:

$$\mathrm{act}_t \;=\; f(\boldsymbol{\theta}_t) - f(\boldsymbol{\theta}_{t+1}). \tag{15}$$

We define and compute the trust-region ratio as the predicted-to-actual loss reduction as:

$$\eta_t \;=\; \frac{\mathrm{act}_t}{\mathrm{pred}_t + \varepsilon}, \qquad \varepsilon > 0. \tag{16}$$

If $\eta_t \approx 1$, the local quadratic model is reliable. When $\eta_t \ll 1$, there is a large disagreement between the predicted and actual reductions, indicating that the curvature information is unreliable, often due to nonconvexity. In this case, the optimizer should switch from the proposed MLBFGS to DADMM. We introduce two thresholds $\beta_1 < \beta_2$ as the decision variable for this switch. If $\eta_t < \beta_1$, the iteration is classified as exhibiting slow convergence, and we switch from the proposed method in (b) to (a) for T^* consecutive iterations.

A hard switch between optimizers can destabilize distributed training, as demonstrated in the simulations presented in Section V. To address this, we introduce a *soft switch* mechanism that smoothly blends the updates of the MLBFGS and DADMM during the T^* transition.

Let α_t denote the blending weight on MLBFGS and $\tau_t = 1-\alpha_t$ the weight on DADMM. We modulate τ_t by the trust-region ratio:

$$\tau_t \;=\; \tau_{\min} \;+\; (\tau_{\max} - \tau_{\min}) \cdot \sigma\big(a\,(\beta_2 - \eta_t)\big), \tag{17}$$

Algorithm 1: Distributed Adaptive Hybrid Optimization (DAHO)

Input: $(\cdot)$, initial parameters $\{\boldsymbol{\theta}_n^0\}_{n\in\mathcal{N}}$, graph G, data D, penalty ρ, thresholds $\beta_1 < \beta_2$, iteration T, transition T^*
Output: $\{\boldsymbol{\theta}_n^T\}_{n\in\mathcal{N}}$

1 **for** $t = 0, 1, \ldots, T-1$ **do**
2 Compute the trust-region ratio η_t using (16);
3 **if** $\eta_t \geq \beta_1$ **then**
   `// Modified second-order method MLBFGS`
4 **for** $n \in \mathcal{N}$ **do**
5 Update dual $\boldsymbol{\Lambda}_n^{t+1}$ using (10);
6 Update primal $\boldsymbol{\theta}_n^{t+1}$ using (13);
7 **else**
8 **for** $t = 0, 1, \ldots, T^*-1$ **do**
   `// Soft switch between MLBFGS and DADMM methods`
9 Compute τ_t using (17);
10 **for** $n \in \mathcal{N}$ **do**
11 Update $\boldsymbol{\theta}_n^{t+1}$ using (18);
   `// ADMM based first-order method DADMM`
12 **for** $n \in \mathcal{N}$ **do**
13 Update $\boldsymbol{p}_n^{t+1}$ using (6);
14 Update $\boldsymbol{\theta}_n^{t+1}$ using (9);
15 **return** $\{\boldsymbol{\theta}_n^T\}_{n\in\mathcal{N}}$

where $\sigma(z) = \frac{1}{1+e^{-z}}$ is the sigmoid function, $a > 0$ controls transition sharpness, $\tau_{\min}$ and $\tau_{\max}$ represent lower and upper bounds of the blending. The parameter updates from MLBFGS and DADMM, denoted by $\mathrm{MLBFGS}(\boldsymbol{\theta}_t)$ and $\mathrm{DADMM}(\boldsymbol{\theta}_t)$, respectively, are blended as:

$$\boldsymbol{\theta}_{t+1} \;=\; \alpha_t \cdot \mathrm{MLBFGS}(\boldsymbol{\theta}_t) \;+\; \tau_t \cdot \mathrm{DADMM}(\boldsymbol{\theta}_t). \tag{18}$$

It can be seen that when $\eta_t \ll \beta_2$, then $\tau_t \to \tau_{\max}$ DADMM dominates, and when $\eta_t \gg \beta_2$, then $\tau_t \to \tau_{\min}$ MLBFGS dominates. The transition between the two methods is governed by changes in the trust-region ratio, enabling a smooth blend according to the convergence performance of MLBFGS.

By applying this adaptive mechanism, (i) automatic switch control to first-order methods is achieved when second-order curvature becomes unreliable, eliminating the need for manual scheduling, and (ii) by dynamically reallocating trust between solvers during the transition, a smooth handover is ensured, thereby stabilizing training.

The proposed DAHO algorithm is shown in Algorithm 1 with modified second-order MLBFGS update from lines 3-7, soft switch from lines 8-11, and ADMM based first-order method DADMM from lines 12-14.

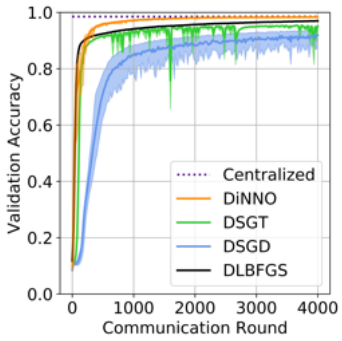
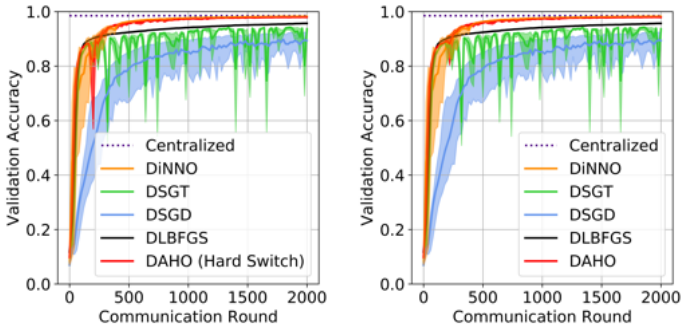


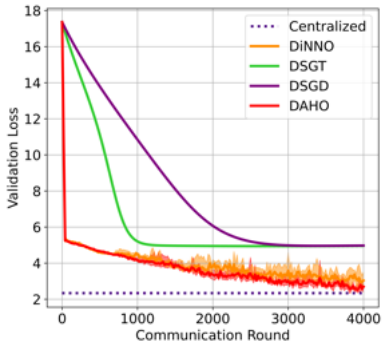
Fig. 2: Validation accuracy as a function of communication rounds for four comparative distributed algorithms, with the average accuracy across 10 robots shown as solid lines.



(a) (b)

Fig. 3: Validation accuracy as a function of communication rounds for DAHO **before** (a) and **after** (b) applying the proposed soft-switch method.

(a)



(b)

Fig. 4: (a) Robot paths overlaid on the ground-truth floor plan. The highlighted path includes the robot's LiDAR scan beams. (b) Validation loss versus communication rounds.

D. Baseline Algorithms

In Section V, we compare our DAHO with three widely used first-order distributed optimization methods, i.e., DiNNO, DSGD, DSGT, and one second-order method, i.e., distributed LBFGS (DLBFGS). In DiNNO, the primal update in (5) is approximated using $K = 5$ iterations, and its convergence to a local optimum has been previously established.

Similar to DiNNO, both DSGD and DSGT allow each robot to update its local weight parameters based on stochastic gradients that are evaluated using the parameters exchanged with neighboring robots. The update of DSGD follows:

$$\theta_n^{t+1} = \sum_{m \in \mathcal{N}} w_{nm}\, \theta_m^t - \alpha^t\, g\big(\theta_n^t\big)\,, \tag{19}$$

where w_{nm} is an entry of the doubly stochastic matrix, whose sparsity pattern matches that of the graph Laplacian of G. α^t is a diminishing step size, and $g(\theta_n^t)$ denotes a stochastic gradient of the local loss $l(\theta_n^t; D_n)$.

The updates in DSGT have the same overall form as those in DSGD, and it introduce an auxiliary variable to track an estimate of the gradient of the joint loss:

$$\theta_n^{t+1} = \sum_{m \in M_n} w_{nm}\big(\theta_m^t - \alpha^t\, y_m^t\big)\,, \tag{20}$$

$$y_n^{t+1} = \sum_{m \in M_n} w_{nm}\, y_m^t + g\big(\theta_n^{t+1}\big) - g\big(\theta_n^t\big)\,. \tag{21}$$

In DLBFGS [20], each robot updates its own weight parameters using gradients and Hessian information computed from parameters shared by other robots. Robot n first computes a local LBFGS direction $p_n^t = -\boldsymbol{H}_n^t g_n^t$ using its approximated Hessian approximation $\boldsymbol{H}_n^t$ and gradient g_n^t, and then communicates this direction with its neighbors to form a summed search direction to reach consensus in the distributed optimization problem:

$$\theta_n^{t+1} = \theta^t + \alpha^t\, p_n^t, \tag{22}$$

$$p_n^{t+1} = \sum_{m \in M_n} w_{nm}\, p_m^t, \tag{23}$$

where α^t is the step size, and w_{nm} denotes an entry of the mixing matrix W. For simplicity, in the experiment in Section V, we take W to be the averaging mixing matrix, which aggregates parameters from all connected robots with equal weights.

V. EXPERIMENT RESULTS

We provide three distributed multi-robot experiments namely distributed MNIST classification, distributed neural implicit mapping, and distributed reinforcement learning inherited from [7] to demonstrate the performance improvement of our proposed DAHO. DAHO is compared against centralized learning, DSGD, DSGT, and DiNNO in all experiments. Due to the computational and memory overhead of DLBFGS, it is included only in the first experiment to illustrate the fast convergence benefits of second-order methods. All baselines are implemented within the same framework, with only the objective function and dataset adapted per task which demonstrate that DAHO integrates as easily as standard optimizers in DNN training while consistently improving distributed learning performance.

A. Distributed MNIST Classification

We evaluate the classic MNIST classification task [27] in a distributed setting, where 10 robots collaboratively train a neural network to recognize handwritten digits. The dataset is partitioned into 10 disjoint subsets, with each robot accessing only one subset. To enforce a fully distributed regime, each robot is assigned samples from a single digit class.

Fig. 2 reports the average best validation accuracy across the 10 robots versus communication rounds for four comparison methods. DiNNO ultimately achieves the highest final accuracy, while DLBFGS exhibits the fastest initial convergence within the first 200 rounds. However, despite its early advantage, the second-order method slows down later and underperforms DiNNO. This is likely due to instability in curvature information in nonconvex objectives, which can lead to suboptimal search directions, as well as the parameter-averaging mechanism similar to DSGD that may degrade performance in heterogeneous data settings.

In Fig. 3, the performance of DAHO with hard switch and after soft switch are shown in Fig. 3(a), and(b), respectively. Without soft switching, noticeable oscillations arise due to differing update dynamics between the two optimization phases. The soft switch significantly reduces these oscillations. A summary of early and late stage performance for DiNNO and DAHO of all three experiments are provided in Table I, with better performance in *green*, showing that DAHO not only converges faster in the initial phase but also slightly improves the final accuracy relative to DiNNO.

Experiment	Rounds / Iterations	Method	Simulation Result
A	150	DiNNO	0.800
		DAHO	0.862
	1600	DiNNO	0.978
		DAHO	0.981
B	150	DiNNO	5.271
		DAHO	5.224
	3000	DiNNO	3.772
		DAHO	3.014
C	3×10^6	DiNNO	37.85
		DAHO	62.35
	8×10^6	DiNNO	324.52
		DAHO	385.28

TABLE I: Comparison of DiNNO and DAHO on three distributed multi-robot experiments: (A) distributed MNIST classification, (B) distributed neural implicit mapping, and (C) distributed reinforcement learning. Experiments A and B are reported at different communication rounds, where the simulation result denotes validation accuracy and validation loss, respectively. Experiment C is reported at different training iterations, where the simulation result denotes the average episode reward.

B. Distributed Neural Implicit Mapping

Neural implicit mapping methods use RGB or LiDAR observations to train neural networks for scene representation and prediction [28], [29]. Given a 3D world coordinate, the corresponding RGB or LiDAR value and the truncated signed distance function (SDF) are predicted. The training objective is typically the sum of the respective L2 losses for RGB and SDF predictions. In a simplified distributed setting adapted from [7], we deploy a team of 7 robots to collaboratively learn the density field of a 2D floorplan, with both data collection and computation distributed across the team. The floorplan is taken from the CubiCasa5K dataset [30]. Each robot traverses a looped path within a subregion of the floorplan using a LiDAR scanner, and all robots collectively learn a global map of the entire floorplan.

Fig. 4(a) shows the ground truth floorplan and the robots' LiDAR scanning paths. As shown in Fig. 4(b), DAHO achieves the lowest validation loss among distributed baselines (DSGD, DSGT, DiNNO) while only DAHO and DiNNO can achieve near centralized performance.

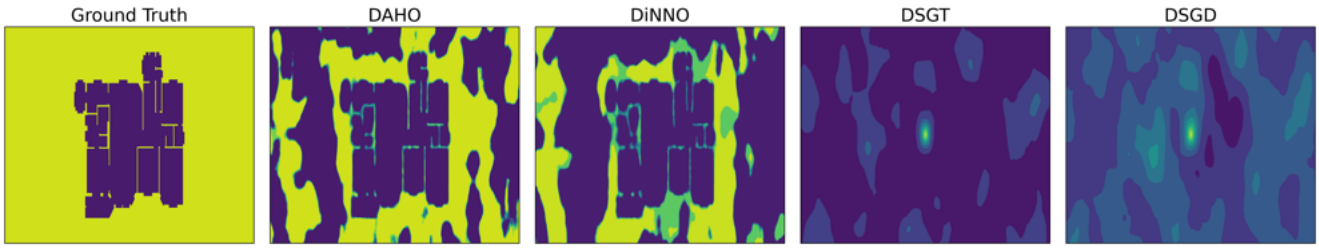
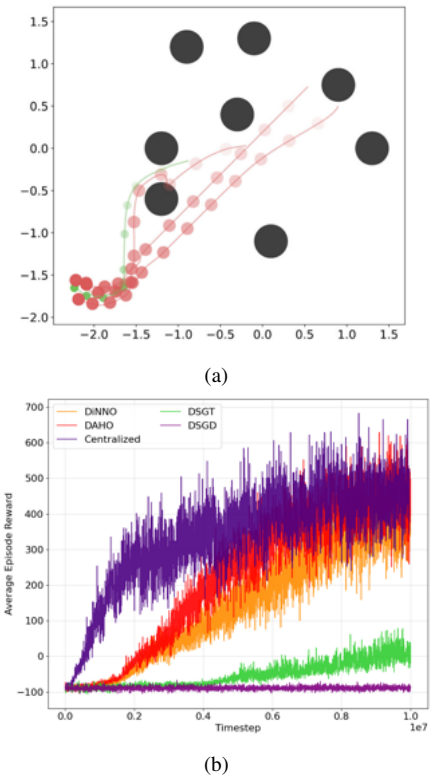


Fig. 5: Mapping reconstructions produced by DAHO and three comparison methods, alongside the ground-truth floor plan.



(a)

(b)

Fig. 6: (a) Environment map for the distributed MARL problem. (b) Average episode reward versus timestep. Only DAHO and DiNNO achieve rewards comparable to the centralized method, indicating successful goal achievement.

Fig. 5 visualizes the reconstructed maps produced by DAHO and the comparison methods. Consistent with the quantitative trends in Fig. 4(b), DAHO yields the most coherent and detailed reconstruction, surpassing DiNNO, while DSGD and DSGT produce fragmented or inconsistent maps. Table I shows that DAHO outperform DiNNO in both early and late phases with lower validation loss.

C. Distributed Reinforcement Learning

Multi-agent reinforcement learning (MARL) [31] is widely used for robotic control, particularly in scenarios where the actions of multiple agents jointly influence a shared environment. We consider a distributed multi-predator–single-prey task [32], where red agents (predators) pursue the green agent (prey) around fixed black obstacles, as illustrated in Fig. 6(a). Each predator observes, acts, and receives rewards sequentially, after which the environment state is updated. The prey follows a heuristic policy, moving in the direction opposite to the nearest predator and the environment is implemented in PettingZoo [33]. The reward function penalizes predators according to their relative distance to the prey and grants a positive reward upon successful capture. A PPO actor–critic framework [34] with distributed optimization is applied.

As shown in Fig. 6(b), DAHO achieves the fastest convergence in terms of average episode reward among distributed baselines and approaches the performance of centralized training. Notably, only DAHO and DiNNO attain near centralized performance, with DAHO reaching strong performance significantly earlier. Table I also proves that DAHO converges faster than DiNNO and also outperforms DiNNO in late phase with higher reward.

VI. CONCLUSIONS

We present DAHO, a distributed adaptive hybrid optimization algorithm that integrates second-order and first-order methods with an adaptive switch mechanism to achieve both rapid convergence and robust performance. DAHO automatically determines the switching criterion based on convergence behavior and employs a soft switch transition policy to ensure stability during the switch. Through extensive multi-robot experiments, we demonstrate that DAHO can be easily applied to distributed learning tasks, similar to standard optimizers, while providing superior efficiency and accuracy.

REFERENCES

[1] J. Hu, M. Mao, H. Bao, G. Zhang, and Z. Cui, "Cp-slam: Collaborative neural point-based slam system," *Advances in Neural Information Processing Systems*, vol. 36, pp. 39 429–39 442, 2023.

[2] O. Walker, F. Vanegas, and F. Gonzalez, "A framework for multi-agent uav exploration and target-finding in gps-denied and partially observable environments," *Sensors*, vol. 20, no. 17, p. 4739, 2020.

[3] B. Wang, L. Zhang, Z. Wang, Y. Zhao, and T. Zhou, "Core: Cooperative reconstruction for multi-agent perception," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 8710–8720.

[4] E. Sebastián, T. Duong, N. Atanasov, E. Montijano, and C. Sagüés, "Physics-informed multi-agent reinforcement learning for distributed multi-robot problems," *IEEE Transactions on Robotics*, 2025.

[5] E. Arnold, M. Dianati, R. De Temple, and S. Fallah, "Cooperative perception for 3d object detection in driving scenarios using infrastructure sensors," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 3, pp. 1852–1864, 2020.

[6] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," *Advances in neural information processing systems*, vol. 30, 2017.

[7] J. Yu, J. A. Vincent, and M. Schwager, "Dinno: Distributed neural network optimization for multi-robot collaborative learning," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1896–1903, 2022.

[8] O. Shorinwa and M. Schwager, "Distributed quasi-newton method for multi-agent optimization," *IEEE Transactions on Signal Processing*, vol. 72, pp. 3535–3546, 2024.

[9] M. Eisen, A. Mokhtari, and A. Ribeiro, "A primal-dual quasi-newton method for exact consensus optimization," *IEEE Transactions on Signal Processing*, vol. 67, no. 23, pp. 5983–5997, 2019.

[10] D. Goldfarb, Y. Ren, and A. Bahamou, "Practical quasi-newton methods for training deep neural networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 2386–2396, 2020.

[11] J. Rafati and R. F. Marica, "Quasi-newton optimization methods for deep learning applications," in *Deep Learning Applications*. Springer, 2020, pp. 9–38.

[12] Z. Jiang, M. Z. Hasan, A. Balu, J. R. Waite, G. Huang, and S. Sarkar, "Fuse: First-order and second-order unified synthesis in stochastic optimization," *arXiv preprint arXiv:2503.04204*, 2025.

[13] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[14] H.-J. M. Shi and D. Mudigere, "Pytorch-lbfgs: A pytorch implementation of l-bfgs," 2018.

[15] L. Bottou, "Stochastic gradient descent tricks," in *Neural networks: tricks of the trade: second edition*. Springer, 2012, pp. 421–436.

[16] A. Bordes, L. Bottou, and P. Gallinari, "Sgd-qn: Careful quasi-newton stochastic gradient descent," *Journal of Machine Learning Research*, vol. 10, pp. 1737–1754, 2009.

[17] J. Gao, X.-W. Liu, Y.-H. Dai, Y. Huang, and J. Gu, "Distributed stochastic gradient tracking methods with momentum acceleration for non-convex optimization," *Computational Optimization and Applications*, vol. 84, no. 2, pp. 531–572, 2023.

[18] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[19] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization," *ACM Transactions on mathematical software (TOMS)*, vol. 23, no. 4, pp. 550–560, 1997.

[20] M. Eisen, A. Mokhtari, and A. Ribeiro, "Decentralized quasi-newton methods," *IEEE Transactions on Signal Processing*, vol. 65, no. 10, pp. 2613–2628, 2017.

[21] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[22] E. Jeong, J. Gwak, T. Kim, and D.-O. Kang, "Distributed deep learning for real-world implicit mapping in multi-robot systems," in *2024 24th International Conference on Control, Automation and Systems (ICCAS)*. IEEE, 2024, pp. 1619–1624.

[23] H. Zhao, B. Ivanovic, and N. Mehr, "Ramen: Real-time asynchronous multi-agent neural implicit mapping," *arXiv preprint arXiv:2502.19592*, 2025.

[24] A. Yahya, A. Li, M. Kalakrishnan, Y. Chebotar, and S. Levine, "Collective robot reinforcement learning with distributed asynchronous guided policy search," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 79–86.

[25] J. E. Dennis, Jr and J. J. Moré, "Quasi-newton methods, motivation and theory," *SIAM review*, vol. 19, no. 1, pp. 46–89, 1977.

[26] A. R. Conn, N. I. Gould, and P. L. Toint, *Trust region methods*. SIAM, 2000.

[27] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[28] E. Sucar, S. Liu, J. Ortiz, and A. J. Davison, "imap: Implicit mapping and positioning in real-time," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 6229–6238.

[29] Y. Deng, Y. Tang, Y. Yang, D. Wang, and Y. Yue, "Macim: Multi-agent collaborative implicit mapping," *IEEE Robotics and Automation Letters*, vol. 9, no. 5, pp. 4369–4376, 2024.

[30] A. Kalervo, J. Ylioinas, M. Häikiö, A. Karhu, and J. Kannala, "Cubicasa5k: A dataset and an improved multi-task model for floorplan image analysis," in *Scandinavian Conference on Image Analysis*. Springer, 2019, pp. 28–40.

[31] H. Zhu, J. Feng, F. Sun, K. Tang, D. Zang, and Q. Kang, "Sharing control knowledge among heterogeneous intersections: a distributed arterial traffic signal coordination method using multi-agent reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, 2025.

[32] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, 2017.

[33] M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. De Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG *et al.*, "Gymnasium: A standard interface for reinforcement learning environments," *arXiv preprint arXiv:2407.17032*, 2024.

[34] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.