Stochastic Unrolled Neural Networks

Samar Hadou[1], Navid NaderiAlizadeh[2], Alejandro Ribeiro[1]
[1]Department of Electrical and Systems Engineering, University of Pennsylvania,
[2] Department of Biostatistics and Bioinformatics, Duke University
{selaraby, aribeiro}@seas.upenn.edu, navid.naderi@duke.edu

This paper develops stochastic unrolled neural networks as learned optimizers for empirical risk minimization (ERM) problems. We view a fixed-depth unrolled architecture as a parameterized optimizer whose layers define a trajectory from an initial random model to a task-specific solution. To handle full datasets, we let each layer interact with randomly drawn mini-batches from the downstream dataset, so that the optimizer incrementally absorbs the entire task. We then train the unrolled optimizer under descent constraints that encourage reductions in loss gradient norms along this trajectory, shaping its dynamics to mimic a convergent stochastic descent method. We prove that such stochastic unrolled networks converge to near-stationary downstream models and quantify performance changes under shifts in the task distribution. As a case study, we instantiate this framework in federated learning by designing an unrolled graph neural network (GNN) architecture derived from decentralized gradient descent, and show that it maintains strong performance under data heterogeneity and asynchronous communication on collaborative image classification tasks.

1. Introduction

Algorithm unrolling has recently emerged as a learning-to-optimize paradigm that unfolds iterative algorithms via learnable neural networks, thereby enabling learning the parameters of the iterative algorithm. The layers of the unrolled architecture correspond to the iterations of the standard algorithm while the outputs of these layers form a trajectory toward the optimal. A key reported advantage of learning these parameters is much faster convergence [1], together with state-of-the-art performance in many applications such as computer vision [2–6], wireless networks [7–12], and medical imaging [13–16], among others [17–22]. Yet, the viewpoint that *learning itself is an optimization process* and that unrolling can shape the learning dynamics has received far less attention.

In this work, we adopt the viewpoint that an unrolled network can serve as a learned optimizer for entire learning problems. Each task in a meta-distribution specifies an empirical risk minimization (ERM) objective over its own dataset, and the unrolled network is trained to map that task to a fitted model in a fixed number of layers. Unlike meta-learning [23, 24], which typically learns an initialization and still relies on a few gradient steps at test time, our goal is to learn an optimizer whose forward pass implements a complete optimization run, producing a task-specific solution without external iterative refinement. Two challenges, however, arise. First, the natural input to an optimizer is a full dataset, while an unrolled network has a fixed, relatively low-dimensional input interface, making it impractical to feed all samples at once. Second, the optimizer must behave reliably across a family of ERM problems, so it is not sufficient to learn a good end-to-end mapping from data to final model parameters. That is, the layerwise dynamics must be stable, progressive, and robust under variations in the task distribution.

To address these challenges, we introduce *stochastic unrolling*, a framework for training unrolled neural networks as learned optimizers in the ERM setting. Rather than feeding an entire downstream dataset into an unrolled architecture with fixed, relatively small input size, we inject independently sampled mini-batches from the dataset at each layer, so that the whole dataset is seen across the depth of the unrolled network. This stochastic feeding scheme makes the layerwise updates resemble stochastic gradient methods as each layer sees only a noisy view of the loss landscape, and the

Third Conference on Parsimony and Learning (CPAL 2026).

resulting descent directions are random. To prevent this stochasticity from undermining convergence and robustness on new tasks, we impose *descent constraints* during training that force the norm of the stochastic gradient of the loss to decrease across layers, effectively training the unrolled architecture to behave as a convergent optimizer. These constraints were reported to improve the robustness of classical unrolled networks [25], and we prove that they provide similar guarantees for stochastic unrolled networks. We theoretically show that such a stochastic unrolled network converges to a near-stationary downstream model and provide bounds that quantify its out-of-distribution (OOD) generalizability under distribution shift in the task distribution.

As a concrete case study, we instantiate stochastic unrolling in federated learning (FL). We consider a vanilla FL setup in which multiple clients collaboratively solve ERM problems over local datasets. We call the resulting FL optimizer Stochastic UnRolled Federated learning, or SURF, realized via a graph neural network (GNN)-based unrolled architecture, termed U-DGD, that unrolls decentralized gradient descent (DGD). In U-DGD, we encode communication and local computation through learnable graph filters and shared fully-connected perceptrons, respectively. This yields an unrolled optimizer that can operate both over general communication graphs (decentralized FL) and star topologies (classical server-based FL). This construction illustrates how stochastic unrolling and descent constraints can be combined with problem-specific algorithmic structure to produce learned optimizers that are both data-driven and endowed with convergence-oriented inductive biases.

In summary, our contributions are as follows:

- We develop *stochastic unrolling*, which enables processing of entire datasets by unrolled networks and thereby extends algorithm unrolling to ERM learning problems.
- We force the unrolled architecture to learn to converge by imposing descent constraints during training, requiring the stochastic gradient norm of the loss to decrease across layers.
- We theoretically prove that a stochastic unrolled network converges to near-stationary downstream models and provide out-of-distribution (OOD) generalization bounds.
- As a case study, we focus on vanilla FL and design U-DGD by unrolling DGD in GNN-based unrolled layers that can handle both decentralized and classical FL problems.

2. Related Work

Learning to Optimize (L2O). Our work is mostly related to the broad research area of L2O [26], which aims to automate the design of optimization methods by training optimizers on a set of training problems. L2O has achieved notable success in many optimization problems including ℓ_1-regularization [27], neural-network training [21, 24], minimax optimization [28], and black-box optimization [29], among many others.

Prior work in L2O can be divided into two categories: model-free and model-based optimizers. Model-free L2O aims to train an iterative update rule that does not take any analytical form and relies mainly on general-purpose recurrent neural network (RNNs) and long short-term memory networks (LSTMs) [24, 29–33]. Model-based L2O, on the other hand, provides compact, interpretable learning networks by leveraging both model-based algorithms and data-driven learning paradigms [27, 34]. As part of this category, algorithm unrolling aims to unroll the hyperparameters of a standard iterative algorithm in a neural network to learn them. The seminal work [27] unrolled the iterative shrinkage thresholding algorithm (ISTA) for sparse coding problems. Following [27], many other algorithms have been unrolled, including, but not limited to, projected gradient descent [35], the primal-dual hybrid gradient algorithm [36, 37], and Frank-Wolfe [38].

Learning to Learn (L2L). L2L refers to frameworks that extend L2O to training neural networks in small data regimes, e.g., few-shot learning [23]. Learning to learn has strong ties to meta-learning, but they differ in their ultimate goal; meta-learning, e.g., model-agnostic meta-learning (MAML) [39], aims to learn an initial model that can be fine-tuned in a few gradient updates, whereas L2L

aims to learn the gradient update and the step size. General purpose LSTM-based models, e.g., [21, 24, 40] are the most popular among L2L models.

Algorithm Unrolling using GNNs. Algorithm unrolling has also been introduced to distributed optimization problems with the help of graph neural networks (GNNs). One of the first distributed algorithms to be unrolled was weighted minimum mean-square error (WMMSE) [41], which benefited many applications including wireless resource allocation [42, 43] and multi-user multiple-input multiple-output (MU-MIMO) communications [9, 44–48]. Several other distributed unrolled networks have been developed for graph topology inference [49], graph signal denoising [50, 51] and computer vision [52], among many others.

3. Stochastic Unrolling

We aim to train an unrolled network $\boldsymbol{\Phi} : \mathcal{T} \to \mathcal{W}$ that acts as a learned optimizer for a family of tasks. A task $\mathbb{T} \in \mathcal{T}$ is a statistical risk minimization (SRM), or equivalently an ERM problem, that fits a downstream model $\mathbf{W}^* \in \mathcal{W}$ to a data distribution $\mathcal{D}$. The model fitting task is with respect to some statistical loss $\widehat{f}$, e.g., regression or entropy, shared among all tasks. Each task is therefore distinct only through the data distribution it fits. The training problem can be cast as

$$\boldsymbol{\Theta}^* \;=\; \underset{\boldsymbol{\Theta}}{\operatorname{argmin}} \;\; \mathbb{E}\Big[\widehat{f}\big(\boldsymbol{\Phi}(\mathcal{D};\boldsymbol{\Theta});\, \mathcal{D}\big)\Big], \tag{1}$$

where $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_l\}_{l=1}^L$ is the unrolled parameterization of the L-layer model $\boldsymbol{\Phi}$ and the expectation is with respect to the task distribution $\mathcal{D}_{\mathcal{T}}$. At inference, the unrolled network predicts a model $\mathbf{W}'$ for an unseen task $\mathbb{T}'$. This setup resembles meta-learning in that we train over a distribution of tasks, but our objective is fundamentally different. Problem (1) does not learn an initialization to be fine-tuned by a hand-designed optimizer in a few gradient steps, as in meta-learning. Instead, it learns the optimizer itself, so that the L layer forward pass of $\boldsymbol{\Phi}$ directly outputs a task specific fitted model for each downstream data distribution.

Two challenges, however, arise in training unrolled networks via (1). First, the unrolled network effectively takes entire data distributions (or datasets) as inputs to predict the fitted models, which becomes increasingly difficult to realize as dataset sizes grow. Second, the training objective in (1) does not *guarantee* that the trained unrolled network behave reliably across a family of ERM problems and it does not provide robustness guarantees under even mild distribution shifts in the task distribution. To this end, we introduce *stochastic unrolled neural networks*, whose layers process randomly drawn mini-batches and are trained under descent constraints, alleviating both challenges.

3.1. Stochastic Unrolled Neural Networks

We devise stochastic unrolled neural networks, whose layers imitate stochastic gradient descent (SGD) over the loss landscape. Rather than feeding an entire downstream dataset into the unrolled architecture at once, we inject independently sampled mini-batches from the dataset at each layer. For a given task determined by a data distribution $\mathcal{D}$, the l-th layer independently samples a mini-batch $\mathbf{B}_l$ and pushes the output of the previous layer $\mathbf{W}_{l-1}$ into a new estimate,

$$\mathbf{W}_l = \Phi_l\big(\mathbf{W}_{l-1}, \mathbf{B}_l; \boldsymbol{\theta}_l\big), \;\; \forall l, \tag{2}$$

where $\mathbf{W}_0$ is a random initialization. In order to ensure that no data point is systematically discarded, we choose the number of layers L and the batch size so that, within a single forward pass, every training sample is seen by at least one layer. The batch size is then held fixed, as it determines the input dimension of each layer.

The sequence of layer outputs $\{\mathbf{W}_l\}_l$ thus defines a trajectory of models, starting from a random initialization $\mathbf{W}_0$ and ending at $\mathbf{W}_L$. To prevent mini-batch stochasticity from undermining convergence, we make descent along this trajectory a feasibility requirement of learning. The training of

stochastic unrolled networks, therefore, becomes the constrained problem,

$$\begin{aligned} \boldsymbol{\Theta}^* \;=\; \underset{\boldsymbol{\Theta}}{\operatorname{argmin}}\; & \mathbb{E}\Big[\widehat{f}\big(\boldsymbol{\Phi}(\mathcal{D};\boldsymbol{\Theta});\,\mathcal{B}\big)\Big] \\ & \text{s.t. } \mathbb{E}\Big[\,\big\|\nabla_{\mathbf{W}}\widehat{f}\big(\mathbf{W}_l;\,\mathbf{B}_l\big)\big\| - (1-\epsilon)\,\big\|\nabla_{\mathbf{W}}\widehat{f}\big(\mathbf{W}_{l-1};\,\mathbf{B}_{l-1}\big)\big\|\,\Big] \leq 0,\;\forall l, \end{aligned} \tag{3}$$

where $\mathcal{B} = \cup_{l=1}^{L} \mathbf{B}_l$ is the set that contains all mini-batches along the forward pass, $\|\cdot\|$ is the Frobenius norm, and $\epsilon \in (0,1)$ is a design parameter. The constraint enforces that, on average across tasks, the norm of the stochastic gradient decreases by at least a factor $(1-\epsilon)$ from one layer to the next, even under the randomness introduced by relying on mini-batches to estimate a descent direction. This makes the learned dynamics mimic a convergent optimization algorithm across the entire task family, rather than merely fitting the final outputs. Since the loss function $\widehat{f}$ is typically non-convex with respect to $\mathbf{W}$, we focus on convergence to local minima. We analyze this convergence guarantees in Section 3.3 after providing a training algorithm for (3).

3.2. Training of Stochastic Unrolled Networks

The formulation in (3) is a nonconvex constrained problem, which is in general difficult to solve. Rather, we resort to its dual problem. The latter is formulated through the Lagrangian function,

$$\mathcal{L}\big(\boldsymbol{\Theta},\boldsymbol{\lambda}\big) = \mathbb{E}\Big[\widehat{f}\big(\boldsymbol{\Phi}(\mathcal{D};\boldsymbol{\Theta});\,\mathcal{B}\big)\Big] + \sum_{l=1}^{L} \lambda_l\, \mathbb{E}\Big[\,\big\|\nabla \widehat{f}\big(\mathbf{W}_l;\,\mathbf{B}_l\big)\big\| - (1-\epsilon)\,\big\|\nabla \widehat{f}\big(\mathbf{W}_{l-1};\,\mathbf{B}_{l-1}\big)\big\|\,\Big], \tag{4}$$

where $\boldsymbol{\lambda} \in \mathbb{R}_+^L$ is a vector collecting the dual variables $\{\lambda_l\}_{l=1}^L$. The dual problem can then be cast as

$$\widehat{D}^* \;=\; \max_{\boldsymbol{\lambda}\in\mathbb{R}_+^L}\; \min_{\boldsymbol{\Theta}}\; \widehat{\mathcal{L}}\big(\boldsymbol{\Theta},\boldsymbol{\lambda}\big), \tag{5}$$

where $\widehat{\mathcal{L}}$ is the empirical Lagrangian, evaluated over Q task instances. The max–min formulation in (5) can be interpreted as solving a family of regularized ERM problems in sequence, each corresponding to a different choice of Lagrange multipliers. In this view, the multipliers play the role of adaptive regularization weights and are updated by projected gradient ascent to maximize the dual function. Solving the dual problem thus amounts to an alternating scheme, which is minimization with respect to $\boldsymbol{\Theta}$ followed by projected gradient ascent in $\boldsymbol{\lambda}$, resulting in the primal–dual procedure summarized in Algorithm 1. As could be envisaged, our training method is agnostic to the underlying task-dependent structure of the stochastic unrolled network.

While classical duality theory [53] affirms that nonconvex constrained problems may exhibit a nonzero duality gap, recent work has shown that training deep neural networks exhibits small duality gap [54]. We capture these observations in the theorem below to keep the subsequent discussion self-contained.

Theorem 1 (Constrained Learning Theorem (informal))**.** *A stationary point of* (5) *is a near-optimal and near-feasible solution to* (3) *under some mild assumptions. That is, for each l,*

$$\mathbb{E}\Big[\,\big\|\nabla_{\mathbf{W}}\widehat{f}\big(\mathbf{W}_l;\,\mathbf{B}_l\big)\big\| - (1-\epsilon)\,\big\|\nabla_{\mathbf{W}}\widehat{f}\big(\mathbf{W}_{l-1};\,\mathbf{B}_{l-1}\big)\big\|\,\Big] \;\leq\; \zeta(Q,\delta),$$

with probability $1-\delta$, and $\zeta(Q,\delta)$ measures the sample complexity.

A formal statement of this theorem is provided in Appendix A. The first implication drawn from Theorem 1 is that (3) has the same sample complexity as (1), and therefore it is equally easy. The second implication is that (5) yields with high probability near-optimal, near-feasible solutions to (3). Consequently, each unrolled layer satisfies the descent constraints up to an error controlled by the sample complexity and takes a step in a stochastic descent direction with probability $1-\delta$. The value δ depends on the number of training tasks Q and can be kept arbitrary small.

Algorithm 1 Primal-dual Training Algorithm

Inputs: number of epochs, step sizes $\eta_1, \eta_2 > 0$
Initialize: $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_l\}_{l=1}^{L}, \boldsymbol{\lambda} = \{\lambda_\ell\}_{\ell=1}^{L}$
for each epoch **do**
    for each batch **do**
        Sample a task $\mathbb{T} \in \mathcal{T}$ with associated data distribution $\mathcal{D}$
        Execute the forward pass $\boldsymbol{\Phi}\big(\mathcal{D}; \boldsymbol{\Theta}\big)$ by feeding a mini-batch $\mathbf{B}_l \sim \mathcal{D}$ to layer l for all l
        Compute the Lagrangian $\widehat{\mathcal{L}}(\boldsymbol{\Theta}, \boldsymbol{\lambda})$ as in (4)
        Primal update: $\boldsymbol{\Theta} = \boldsymbol{\Theta} - \eta_1 \nabla_{\boldsymbol{\Theta}} \widehat{\mathcal{L}}(\boldsymbol{\Theta}, \boldsymbol{\lambda})$
        Dual update: $\boldsymbol{\lambda} = \Big[\boldsymbol{\lambda} + \eta_2 \nabla_{\boldsymbol{\lambda}} \widehat{\mathcal{L}}(\boldsymbol{\Theta}, \boldsymbol{\lambda})\Big]_+$
Return $\boldsymbol{\Theta}^* \leftarrow \boldsymbol{\Theta}$.

3.3. Convergence Guarantees and OOD Generalizability

Theorem 1 is not, by itself, sufficient to obtain convergence guarantees. Its conclusion is that the trained network satisfies each constraint with probability $1-\delta$, even though (3) requires each layer to enforce descent. The probability that all L constraints are satisfied is then $(1-\delta)^L$, which can become very small when L is large. As a result, Theorem 1 does not directly ensure the convergence behavior we ultimately seek. In this section, we track the minimum gradient norm along the trajectory in Theorem 2 and use a supermartingale argument to show that this minimum is eventually small and controllable. Finally, we show in Theorem 3 that this neighborhood degrades gracefully under shifts in the task distribution.

Theorem 2. *Assume that the loss function $\widehat{f}(\mathbf{W})$ is M-Lipschitz and consider a sequence of $\{\mathbf{W}_l\}_l$ generated by a stochastic unrolled network trained via Algorithm 1. Then, it holds that*

$$\lim_{l\to\infty} \mathbb{E}\Big[\min_{k\leq l} \big\|\nabla \widehat{f}\big(\mathbf{W}_k;\, \mathbf{B}_k\big)\big\|\Big] \;\leq\; \frac{1}{\epsilon}\left(\zeta(Q,\delta) + \frac{\delta M}{1-\delta}\right) \quad a.s. \tag{6}$$

The detailed proof of Theorem 2 is relegated to Appendix B.1. Theorem 2 confirms that the stochastic unrolled network is guaranteed to converge to a region around a local minimum. The size of this region depends on the sample complexity, the Lipschitz constant of the loss function and its gradient, and lastly a design parameter ϵ of the imposed constraints. The larger ϵ, which is equivalent to imposing an aggressive reduction on the gradients, the closer we are guaranteed to get to a local optimal $\mathbf{W}^*$. Moreover, the sample complexity controls the failure probability δ, and for sufficiently large Q, we can keep δ arbitrary small and eliminate the second term of the bound.

Driving the gradient norm to a small value only certifies approximate stationarity and does not automatically ensure convergence to a *good* minimum. The quality of the attained stationary point can depend on additional factors such as the size of the downstream dataset associated with each task and the expressivity of the function class $\mathcal{W}$. To keep the presentation focused, we adopt the standard assumption that datasets are sufficiently large and models sufficiently expressive so that approximate stationarity corresponds to near-optimal solutions for the tasks of interest.

Finally, we analyze how the descent constraints behave when the task distribution shifts. Let $D_{\mathcal{T}}$ denote the training task distribution and $D'_{\mathcal{T}}$ a shifted distribution. We consider the stochastic unrolled optimizer trained on tasks from $D_{\mathcal{T}}$ and apply it to tasks drawn from $D'_{\mathcal{T}}$. The next lemma shows that, under this shift, the layerwise descent constraints remain approximately satisfied, with a violation that grows at most linearly with the distance $d(D_{\mathcal{T}}, D'_{\mathcal{T}})$.

Lemma 1. *Under Assumption 6 (Appendix B.2), a stochastic unrolled network trained over a distribution $\mathcal{D}_{\mathcal{T}}$ generates when executed on an OOD distribution $\mathcal{D}'_{\mathcal{T}}$ a sequence of layer outputs $\{\mathbf{W}_l\}_l$ that satisfies*

$$\mathbb{E}_{\mathcal{D}'_{\mathcal{T}}}\Big[\, \big\|\nabla_{\mathbf{W}} \widehat{f}\big(\mathbf{W}_l;\, \mathbf{B}_l\big)\big\| - (1-\epsilon)\, \big\|\nabla_{\mathbf{W}} \widehat{f}\big(\mathbf{W}_{l-1};\, \mathbf{B}_{l-1}\big)\big\| \,\Big] \;\leq\; \zeta(Q,\delta) + 2M\, d(\mathcal{D}_{\mathcal{T}}, \mathcal{D}'_{\mathcal{T}}), \tag{7}$$

with probability $1-\delta$ each, where $d(\cdot,\cdot)$ is a bounded symmetric distance metric.

Lemma 1, therefore, guarantees that, even under task distribution shift, each layer update remains an approximate descent step, with the deviation from ideal descent controlled by $d(D_{\mathcal{T}}, D'_{\mathcal{T}})$. As a consequence, the learned optimizer retains stable, approximately descending dynamics under task-level distribution shift, and its trajectory converges to a neighborhood of a stationary point whose size depends on this term, as formalized in the next theorem.

Theorem 3. *The sequence of layer outputs $\{\mathbf{W}_l\}_l$ generated according to Lemma 1 also satisfies*

$$\lim_{l\to\infty} \mathbb{E}_{\mathcal{D}'_{\mathcal{T}}}\Big[\min_{k\le l} \big\|\nabla \widehat{f}\big(\mathbf{W}_k;\, \mathbf{B}_k\big)\big\|\Big] \;\le\; \frac{1}{\epsilon}\Big(\zeta(Q,\delta) + 2Md(\mathcal{D}_{\mathcal{T}}, \mathcal{D}'_{\mathcal{T}}) + \frac{\delta M}{1-\delta}\Big). \tag{8}$$

4. Stochastic Unrolled Federated Learning

As a case study, we consider a vanilla federated learning (FL) scenario, in which n clients coordinate to fit a model $\mathbf{w}$ to a data distribution $\mathcal{D}$. Formally, the federated problem can be expressed as

$$P^* \;=\; \min_{\mathbf{w}_1,\dots,\mathbf{w}_n} \frac{1}{n}\sum_{i=1}^{n} \widehat{f}_i(\mathbf{w}_i), \quad \text{s.t.} \qquad \mathbf{w}_i = \frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i} \mathbf{w}_j, \quad \forall i \in \mathcal{V}, \tag{9}$$

where $\mathbf{w}_i$ is a *local* version of the global variable $\mathbf{w}$ stored at agent i, and all $\mathbf{w}_i$'s are arranged in the rows of the matrix $\mathbf{W}$. The local objective $\widehat{f}_i(\mathbf{w}_i)$ fits a model $\mathbf{w}_i$ to the local data distribution $\mathcal{D}_i$ while the constraints mandate that each local model remains equal to the average of the direct neighbors' models. When satisfied, these constraints boil down to constraints of the form $\mathbf{w}_i = \mathbf{w}_j$ for all i and j, hence leading to consensus among agents.

It is customary to solve the decentralized problem in (9) using the DGD algorithm, which relies on limited communication between agents. At each iteration l, DGD performs the update rule:

$$\mathbf{w}_i(l) \;=\; \sum_{j\in\mathcal{N}_i\cup\{i\}} \alpha_{ij}\mathbf{w}_j(l-1) - \beta\nabla\widehat{f}_i(\mathbf{w}_i(l-1)),\ \forall i, \tag{10}$$

where β is a fixed step size and $\alpha_{ij} = \alpha_{ji}$. The weights α_{ij} are chosen such that $\sum_{j=1}^{n}\alpha_{ij} = 1$ for all i to ensure that (10) converges [55]. The update rule in (10) can be interpreted as letting the agents descend in the opposite direction of the local gradient $\nabla f_i(\mathbf{w}_i(l-1))$ as they move away from the (weighted) average of their neighbors' estimates $\mathbf{w}_j(l-1)$. Each iteration can then be divided into two steps; first the agents aggregate information from their direct neighbors and then they update their weights based on the gradient of their local objective functions.

4.1. GNN-based Unrolled DGD

We design a graph neural network (GNN) architecture as the unrolled parameterization $\boldsymbol{\Theta}$, whose layers are crafted to mimic the DGD update in (10). Concretely, we implement each iteration of (10) using a combination of graph filters for neighbor aggregation and single-layer fully connected perceptrons for local updates. We further distinguish between two settings: (i) decentralized FL over a general communication graph, and (ii) classical FL over a star topology with a central server.

Decentralized FL. We unfold the update rule of DGD in a learnable neural layer of the form

$$\mathbf{w}_{i,l} \;=\; \big[\mathbf{H}_l(\mathbf{W}_{l-1})\big]_i - \sigma\Big(\mathbf{M}_l\,[\mathbf{w}_{i,l-1}\|\mathbf{b}_{i,l}] + \mathbf{d}_l\Big), \tag{11}$$

where $[.]_i$ refers to the i-th row of a matrix, $\|$ denotes a vertical vector concatenation, $\mathbf{b}_{i,l}$ is a vectorization of the mini-batch sampled by agent i at layer l, and σ is a non-linear activation function. In (11), we replace the first term in (10) with a learnable graph filter (see (12)) and the second term with a single fully-connected perceptron. The learnable parameters are then $\boldsymbol{\theta} = \{\mathbf{h}_l, \mathbf{M}_l, \mathbf{d}_l\}$, which we elaborate on in the rest of this subsection.

The graph filter, the building block of GNNs [56], aggregates information from up to K-hop neighbors,

$$\mathbf{H}(\mathbf{W}_{l-1}) = \sum_{k=0}^{K} h_{k,l}\mathbf{S}^k\mathbf{W}_{l-1}, \tag{12}$$

where $\mathbf{S}$ is a graph shift operator, e.g., graph adjacency or Laplacian. The graph filter executes a linear combination of information gathered from up to K-hop neighbors, and, in turn, requires K communication rounds. Here, the filter coefficients $\mathbf{h}_l = \{h_{k,l}\}_{k=0}^{K}$ that weigh the information aggregated from different hop neighbors are the learnable parameters. Equation (12) and the first term of (10) are essentially the same when K is set to 1 and h_k to 1 for all k and $\mathbf{S}$ is chosen to be the (normalized) graph adjacency matrix. In U-DGD, however, the goal is to learn the weights $h_{k,l}$ to accelerate the unrolled network's convergence.

The other component of (11) is a single-layer fully-connected perceptron, which is implemented locally and whose weights $\mathbf{M}_l$ and $\mathbf{d}_l$ are shared among all the agents. The role of this neural perceptron is to perform the local update of the weights once every K communication rounds. The input to this perceptron at each agent is the previous estimate $\mathbf{w}_{i,l-1} \in \mathbb{R}^d$ concatenated with a mini-batch $\mathbf{b}_{i,l}$. Each batch is a concatenation of the sampled data points, where the input data and label of one example follow each other. Since the parameters of the fully-connected perceptron are shared between all the agents, U-DGD learners inherit the permutation equivariance of graph filters and graph neural networks, as well as transferability to graphs with different sizes and stability to small graph perturbations [57, 58].

Classical FL. Classical FL can be retrieved from (9) when the underlying graph is a star topology. In such a topology, the nodes only communicate with one another through a central node, which can function as a server in classical FL settings. Therefore, U-DGD can extend to classical FL with a minor adjustment to account for the fact that the server does not possess local data.

The unrolled layer of the server, denoted as node 0, employs only a graph filter to aggregate information only from their direct neighbors, i.e.,

$$\mathbf{w}_{0,l} = h_l[\mathbf{S}\mathbf{W}_{l-1}]_0. \tag{13}$$

The fully-connected perceptron in (11) is excluded here since the server node does not locally execute a weight update. Furthermore, (13) is identical to the first row of (12) when K is 1 and $k = 0$ is omitted. This means that the server only performs a weighted average of the information received from the other nodes based on the values at the first row of $\mathbf{S}$. The rest of the nodes with indices $i > 0$, on the other hand, apply the exact form of (11) while constraining K to 1.

4.2. Numerical Experiments

We run experiments to show the convergence rate of SURF under different settings.[1]

Set-up. We consider a network of $n = 100$ agents, which collaborate to train a softmax layer of an image classifier. The softmax layer is fed by the outputs of the convolutional layers of a ResNet18 backbone, whose weights are pre-trained and kept frozen during the training process. To train a U-DGD network, we consider a meta-training dataset, which consists of 600 class-imbalanced datasets. Each dataset has a different label distribution and contains $6,000$ images (that is, 45 training examples/agent and 15 for testing) that are evenly divided between the agents. The training details are relegated to Appendix C.

Convergence Guarantees. To assess the effects of the descent constraints on the training performance, we compare the test loss and accuracy with and without these constraints in Figure 1. The network among the agents is chosen here as a 3-degree regular graph. The figure shows that SURF, depicted in blue, converges gradually to the optimal loss/accuracy over the layers. However, the standard unrolled optimizer trained without the descent constraints failed to maintain a similar behavior even though it achieves the same performance at the final layer. In fact, the accuracy jumps from 0% to 96% at the last layer, which would make the optimizer more vulnerable to additive noise and perturbations in the layers' inputs, as we show in the following experiment.

Decentralized FL. We run experiments with two arbitrary graph topologies, specifically 3-degree regular graphs and random graphs, to create decentralized FL environments. In the former topology,

[1]Our code is available at: `https://github.com/SMRhadou/fed-SURF`

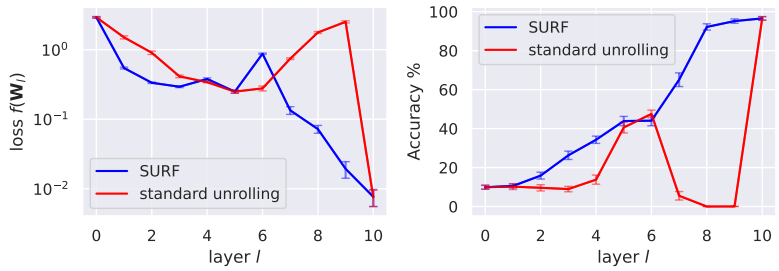

Figure 1: **Convergence Guarantees.** Evolution of the loss and accuracy (evaluated over 30 test datasets sampled from MNIST) across the unrolled layers of U-DGD, trained with and without the descent constraints. Observe that constrained U-DGD (SURF) converges gradually to the optimal.

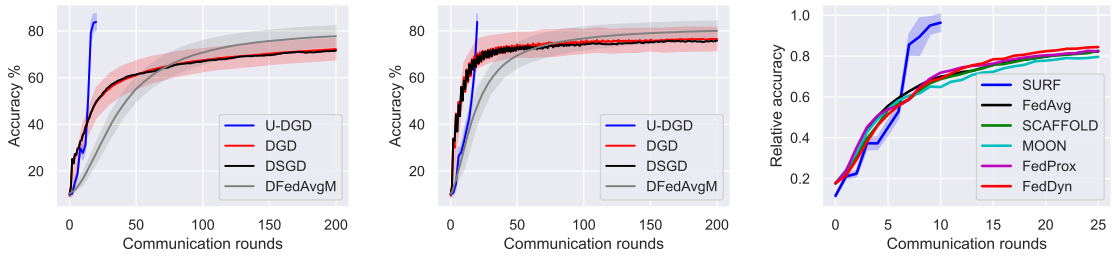

Figure 2: **Convergence rate under different topologies.** Comparisons between the accuracy of U-DGD and FL benchmarks for both i) decentralized FL over 3-degree regular graphs (left) and random graphs (middle), and ii) classical FL with a star graph (right). U-DGD achieves faster convergence rates in all settings surpassing both decentralized and centralized FL methods.

each node connects to exactly 3 other nodes, while in the latter, an edge is connected between any two nodes with probability $p = 0.1$. For both scenarios, we train a U-DGD that consists of 10 unrolled layers, each of which employs a graph filter that aggregates information from up to two neighbors (i.e., $K = 2$). This results in a total of 20 communication rounds between the agents.

We compare the accuracy of U-DGD to other decentralized FL benchmarks: DGD in (10), distributed stochastic gradient descent (DSGD), and decentralized federated averaging (DFedAvgM) [59]. Figure 2 shows that U-DGD exhibits a faster convergence rate as it takes only 20 communication rounds compared to the 200 communication rounds required by the others. This behavior is consistent with U-DGD acting as an accelerated method, and should not be interpreted as diminishing the quality of the baseline algorithms.

Classical FL over a star graph. In addition, we consider a star graph with one of the nodes acting as a server. We set $\mu_\theta = 10^{-3}$, $\epsilon = 0.1$, and $K = 1$ while the rest of the parameters are kept the same as they were in the decentralized experiment. We compare the convergence rate of U-DGD with other FL benchmarks: FedAvg [60], SCAFFOLD [61], MOON [62], FedProx [63], and FedDyn [64]. For fair comparisons, all the methods, including U-DGD, let only 10 agents to share their updated weights with the server node at each communication round. Figure 2 (right) shows that U-DGD has a notably faster convergence than all the benchmarks. The figure also shows the relative accuracy of these methods compared to centralized training. It suggests that SURF almost achieves the same performance attained by central training in 10 communication rounds while the other benchmarks need 25 rounds to reach almost 80% of the centralized performance.

Heterogeneous settings. We evaluate U-DGD on a network of heterogeneous agents that sample their data according to a Dirichlet distribution with concentration parameter α, where smaller values of α indicate greater heterogeneity among agents. The agents are connected through a 3-regular graph. For each value of α, we generate 30 heterogeneous downstream datasets and report the

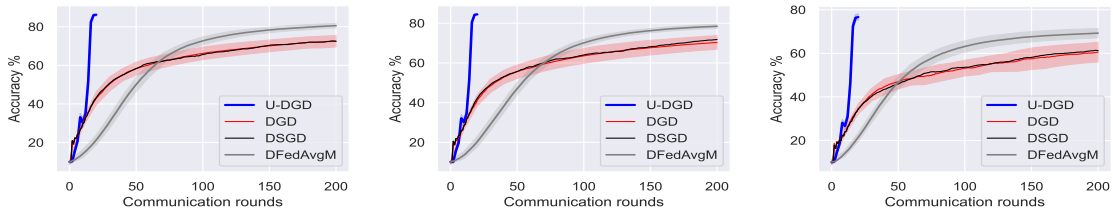

Figure 3: **Heterogeneous settings.** Comparisons between the accuracy of U-DGD and other decentralized benchmarks, evaluated over 30 class-imbalanced CIFAR10 datasets sampled according to a Dirichlet distribution with a concentration parameter (Left) $\alpha = 1$, (Middle) $\alpha = 0.7$, and (Right) $\alpha = 0.3$. The lower α is, the more heterogeneous the agents are.

average accuracy in Figure 3. For the baselines, we retrain the downstream models from scratch for each new α using the same step size (i.e., without re-tuning hyperparameters). In contrast, U-DGD is kept fixed and only executed on the new datasets, without any retraining. The plots show that the performance of all methods deteriorates as α decreases. However, the degradation for U-DGD is noticeably slower than for the baselines, indicating stronger OOD generalization of our descent-constrained optimizer to new downstream tasks with increasingly shifted label distributions. This behavior is consistent with the theory in Section 3.3.

Asynchronous Communications. We also evaluate U-DGD, trained with descent constraints, under perturbations induced by asynchronous communication. In this setting, at each layer n_{asyn} randomly chosen agents fail to update and send their estimates in sync with the rest of the network, so their neighbors instead use outdated versions communicated at previous layers. This induces a shift in the input distribution at each layer, which can degrade the performance of the neural optimizers. We compare the constrained U-DGD (SURF) to its unconstrained counterpart in this asynchronous regime and report their performance in Figure 4. The figure shows that SURF is noticeably more resilient. Its performance deteriorates more slowly than that of the unconstrained U-DGD as n_{asyn} increases. This supports our central claim that enforcing descent constraints at training time is what makes the learned optimizer robust to distribution shifts in its inputs, whether they arise from heterogeneous data or asynchronous communications.

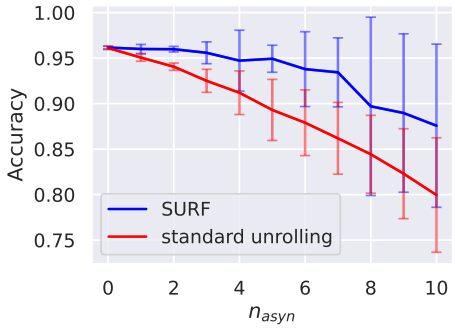

Figure 4: **Asynchronous Communications.** Comparison of the test loss and accuracy in different communication environments where n_{asyn} agents are asynchronous with the rest of the network.

Sensitivity Analysis on ϵ. We conduct a sensitivity analysis on the constraint parameter ϵ. As shown in Figure 5, the choice of the parameter does not affect the last-layer performance. Its primary effect is on the layerwise convergence profile. For very small values of ϵ, intermediate-layer accuracy remains near zero before jumping sharply at the final layer. As ϵ increases, convergence becomes progressively faster, yielding trajectories with steeper and more monotone improvements across layers. For larger values, however, the descent constraints become more stringent, and we observe more frequent constraint violations.

5. Conclusions

In this paper, we introduced a general framework of stochastic unrolled neural networks as learned optimizers for empirical risk minimization. We treat the layers of an unrolled architecture as the steps of a stochastic update procedure that interacts with a task through mini-batches, and by training under constraints that encourage progressive reduction of the loss gradient norm along the trajectory.

Our theoretical analysis shows that such stochastic unrolled optimizers converge to near-stationary downstream models and that their behavior under task distribution shift can be quantified through bounds on the size of the region they approach, which expands gracefully as the source and target distributions diverge.

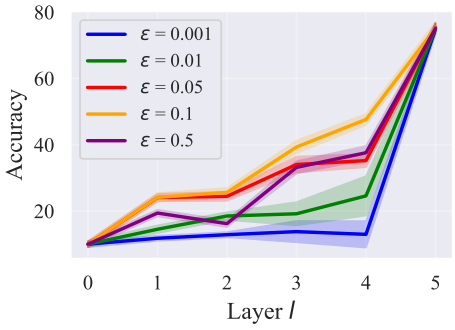

Figure 5: **Sensitivity Analysis on** ϵ for the case with a Dirichlet concentration parameter $\alpha = 0.3$. The same pattern is observed for other values of α.

To make these ideas concrete, we specialized the framework to FL and designed U-DGD, a GNN-based unrolled architecture that mirrors decentralized gradient descent. Experiments on collaborative image classification tasks demonstrate that U-DGD attains target accuracies in substantially fewer communication rounds than standard baselines, and that the constraint-shaped training improves robustness to both data heterogeneity and asynchronous communication. These results suggest that stochastic unrolling, combined with principled control of the optimization dynamics, offers a promising recipe for designing learned optimizers that are fast and resilient across families of tasks.

References

[1] Vishal Monga, Yuelong Li, and Yonina C. Eldar. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *IEEE Signal Processing Magazine*, 38(2):18–44, March 2021.

[2] Kai Zhang, Luc Van Gool, and Radu Timofte. Deep unfolding network for image super-resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3217–3226, 2020.

[3] Xinyi Wei, Hans van Gorp, Lizeth Gonzalez-Carabarin, Daniel Freedman, Yonina C. Eldar, and Ruud J. G. van Sloun. Deep unfolding with normalizing flow priors for inverse problems. *IEEE Transactions on Signal Processing*, 70:2962–2971, 2022. doi: 10.1109/TSP.2022.3179807.

[4] Chong Mou, Qian Wang, and Jian Zhang. Deep generalized unfolding networks for image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17399–17410, 2022.

[5] Yuelong Li, Mohammad Tofighi, Junyi Geng, Vishal Monga, and Yonina C. Eldar. Efficient and interpretable deep blind image deblurring via algorithm unrolling. *IEEE Transactions on Computational Imaging*, 6:666–681, 2020. doi: 10.1109/TCI.2020.2964202.

[6] Peng Qiao, Sidun Liu, Tao Sun, Ke Yang, and Yong Dou. Towards vision transformer unrolling fixed-point algorithm: a case study on image restoration. *arXiv preprint arXiv:2301.12332*, 2023.

[7] Qiyu Hu, Yunlong Cai, Qingjiang Shi, Kaidi Xu, Guanding Yu, and Zhi Ding. Iterative algorithm induced deep-unfolding neural networks: Precoding design for multiuser MIMO systems. *IEEE Transactions on Wireless Communications*, 20(2):1394–1410, 2020.

[8] Arindam Chowdhury, Gunjan Verma, Chirag Rao, Ananthram Swami, and Santiago Segarra. Unfolding WMMSE using graph neural networks for efficient power allocation. *IEEE Transactions on Wireless Communications*, 20(9):6004–6017, 2021.

[9] Lukas Schynol and Marius Pesavento. Coordinated sum-rate maximization in multicell mu-mimo with deep unrolling. *IEEE Journal on Selected Areas in Communications*, 41(4):1120–1134, 2023.

[10] Hao Huang, Yun Lin, Guan Gui, Haris Gacanin, Hikmet Sari, and Fumiyuki Adachi. Regularization strategy aided robust unsupervised learning for wireless resource allocation. *IEEE Transactions on Vehicular Technology*, 2023.

[11] Hao Yang, Nan Cheng, Ruijin Sun, Wei Quan, Rong Chai, Khalid Aldubaikhy, Abdullah Alqasir, and Xuemin Shen. Knowledge-driven resource allocation for wireless networks: A wmmse unrolled graph neural network approach. *IEEE Internet of Things Journal*, 11(10):18902–18916, 2024.

[12] Jianjun Zhang, Christos Masouros, Fan Liu, Yongming Huang, and A. Lee Swindlehurst. Low-complexity joint radar-communication beamforming: From optimization to deep unfolding. *IEEE Journal of Selected Topics in Signal Processing*, pages 1–16, 2025. doi: 10.1109/JSTSP.2024.3522787.

[13] Ping Wang, Lishun Wang, Gang Qu, Xiaodong Wang, Yulun Zhang, and Xin Yuan. Proximal algorithm unrolling: Flexible and efficient reconstruction networks for single-pixel imaging. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 411–421, 2025.

[14] Yuelong Li, Or Bar-Shira, Vishal Monga, and Yonina C Eldar. Deep algorithm unrolling for biomedical imaging. *arXiv preprint arXiv:2108.06637*, 2021.

[15] Nishith Chennakeshava, Tristan SW Stevens, Frederik J de Bruijn, Andrew Hancock, Martin Pekař, Yonina C Eldar, Massimo Mischi, and Ruud JG van Sloun. Deep proximal unfolding for image recovery from under-sampled channel data in intravascular ultrasound. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1221–1225. IEEE, 2022.

[16] Hong Wang, Yuexiang Li, Haimiao Zhang, Deyu Meng, and Yefeng Zheng. InDuDoNet+: A deep unfolding dual domain network for metal artifact reduction in ct images. *Medical Image Analysis*, 85:102729, 2023.

[17] John R Hershey, Jonathan Le Roux, and Felix Weninger. Deep unfolding: Model-based inspiration of novel deep architectures. *arXiv preprint arXiv:1409.2574*, 2014.

[18] Rami Nasser, Yonina C Eldar, and Roded Sharan. Deep unfolding for non-negative matrix factorization with application to mutational signature analysis. *Journal of Computational Biology*, 29(1):45–55, 2022.

[19] Chengen Liu, Geert Leus, and Elvin Isufi. Unrolling of simplicial elasticnet for edge flow signal reconstruction. *IEEE Open Journal of Signal Processing*, pages 1–9, 2023.

[20] Nawel Arab, Yassine Mhiri, Isabelle Vin, Mohammed Nabil El Korso, and Pascal Larzabal. Unrolled expectation maximization algorithm for radio interferometric imaging in presence of non gaussian interferences. *Signal Processing*, page 110035, 2025.

[21] Sachin Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2016.

[22] Samar Hadou and Alejandro Ribeiro. Unrolled graph neural networks for constrained optimization, 2025. URL `https://arxiv.org/abs/2509.17156`.

[23] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Meta-dataset: A dataset of datasets for learning to learn from few examples. In *International Conference on Learning Representations*, 2020.

[24] Marcin Andrychowicz, Misha Denil, Sergio Gómez Colmenarejo, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, page 3988–3996, 2016.

[25] Samar Hadou, Navid NaderiAlizadeh, and Alejandro Ribeiro. Robust stochastically-descending unrolled networks. *arXiv preprint arXiv:2312.15788*, 2023.

[26] Tianlong Chen, Xiaohan Chen, Wuyang Chen, Howard Heaton, Jialin Liu, Zhangyang Wang, and Wotao Yin. Learning to optimize: A primer and a benchmark, July 2021.

[27] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on Machine Learning*, ICML'10, page 399–406, 2010.

[28] Jiayi Shen, Xiaohan Chen, Howard Heaton, Tianlong Chen, Jialin Liu, Wotao Yin, and Zhangyang Wang. Learning a minimax optimizer: A pilot study. In *International Conference on Learning Representations*, 2021.

[29] Yutian Chen, Matthew W. Hoffman, Sergio Gómez Colmenarejo, Misha Denil, Timothy P. Lillicrap, Matt Botvinick, and Nando de Freitas. Learning to learn without gradient descent by gradient descent. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 748–756. PMLR, 06–11 Aug 2017.

[30] Kaifeng Lyu, Shunhua Jiang, and Jian Li. Learning gradient descent: Better generalization and longer horizons. In *International Conference on Machine Learning*, 2017.

[31] Olga Wichrowska, Niru Maheswaranathan, Matthew W. Hoffman, Sergio Gómez Colmenarejo, Misha Denil, Nando de Freitas, and Jascha Sohl-Dickstein. Learned optimizers that scale and generalize. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3751–3760, 2017.

[32] Yuanhao Xiong and Cho-Jui Hsieh. Improved adversarial training via learned optimizer. In *Computer Vision – ECCV 2020*, pages 85–100, 2020.

[33] Haoming Jiang, Zhehui Chen, Yuyang Shi, Bo Dai, and Tuo Zhao. Learning to defend by learning to attack. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 577–585. PMLR, 13–15 Apr 2021.

[34] Daniel Greenfeld, Meirav Galun, Ronen Basri, Irad Yavneh, and Ron Kimmel. Learning to optimize multigrid PDE solvers. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2415–2423, 09-15 Jun 2019.

[35] Raja Giryes, Yonina C. Eldar, Alex M. Bronstein, and Guillermo Sapiro. Tradeoffs between convergence speed and reconstruction accuracy in inverse problems. *Transaction in Signal Processing*, 66(7):1676–1690, apr 2018.

[36] Mingyuan Jiu and Nelly Pustelnik. A deep primal-dual proximal network for image restoration. *IEEE Journal of Selected Topics in Signal Processing*, 15:190–203, 2020.

[37] Jing Cheng, Haifeng Wang, Leslie Ying, and Dong Liang. Model learning: Primal dual networks for fast mr imaging. In *Medical Image Computing and Computer Assisted Intervention MICCAI 2019*, pages 21–29, 2019.

[38] Dong Liu, Ke Sun, Zhangyang Wang, Runsheng Liu, and Zheng-Jun Zha. Frank-wolfe network: An interpretable deep structure for non-sparse coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(9):3068–3080, 2019.

[39] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 1126–1135, August 2017.

[40] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-SGD: Learning to learn quickly for few shot learning. *ArXiv*, abs/1707.09835, 2017.

[41] Qingjiang Shi, Meisam Razaviyayn, Zhi-Quan Luo, and Chen He. An iteratively weighted mmse approach to distributed sum-utility maximization for a mimo interfering broadcast channel. *IEEE Transactions on Signal Processing*, 59(9):4331–4340, 2011.

[42] Arindam Chowdhury, Gunjan Verma, Chirag Rao, Ananthram Swami, and Santiago Segarra. Unfolding wmmse using graph neural networks for efficient power allocation. *IEEE Transactions on Wireless Communications*, 20(9):6004–6017, 2021.

[43] Boning Li, Ananthram Swami, and Santiago Segarra. Power allocation for wireless federated learning using graph neural networks. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5243–5247, 2022.

[44] Qiyu Hu, Yunlong Cai, Qingjiang Shi, Kaidi Xu, Guanding Yu, and Zhi Ding. Iterative algorithm induced deep-unfolding neural networks: Precoding design for multiuser mimo systems. *IEEE Transactions on Wireless Communications*, 20(2):1394–1410, 2021.

[45] Ningxin Zhou, Zheng Wang, Lanxin He, and Yang Huang. A new low-complexity wmmse algorithm for downlink massive mimo systems. In *2022 14th International Conference on Wireless Communications and Signal Processing (WCSP)*, pages 1096–1101, 2022.

[46] Yifan Ma, Xianghao Yu, Jun Zhang, S.H. Song, and Khaled B. Letaief. Augmented deep unfolding for downlink beamforming in multi-cell massive mimo with limited feedback. In *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, pages 1721–1726, 2022.

[47] Lissy Pellaco and Joakim Jaldén. A matrix-inverse-free implementation of the mu-mimo wmmse beamforming algorithm. *IEEE Transactions on Signal Processing*, 70:6360–6375, 2022.

[48] Lukas Schynol and Marius Pesavento. Deep unfolding in multicell mu-mimo. In *2022 30th European Signal Processing Conference (EUSIPCO)*, pages 1631–1635, 2022.

[49] Xingyue Pu, Tianyue Cao, Xiaoyun Zhang, Xiaowen Dong, and Siheng Chen. Learning to learn graph topologies. In *Advances in Neural Information Processing Systems*, 2021.

[50] Siheng Chen, Yonina C. Eldar, and Lingxiao Zhao. Graph unrolling networks: Interpretable neural networks for graph signal denoising. *IEEE Transactions on Signal Processing*, 69:3699–3713, 2021.

[51] Masatoshi Nagahama, Koki Yamada, Yuichi Tanaka, Stanley H. Chan, and Yonina C. Eldar. Graph signal denoising using nested-structured deep algorithm unrolling. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5280–5284, 2021.

[52] Xin Lin, Changxing Ding, Jing Zhang, Yibing Zhan, and Dacheng Tao. Ru-net: regularized unrolling network for scene graph generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19457–19466, 2022.

[53] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[54] Luiz FO Chamon, Santiago Paternain, Miguel Calvo-Fullana, and Alejandro Ribeiro. Constrained learning with non-convex losses. *IEEE Transactions on Information Theory*, 2022.

[55] Angelia Nedic and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, January 2009.

[56] Fernando Gama, Elvin Isufi, Geert Leus, and Alejandro Ribeiro. Graphs, convolutions, and neural networks: From graph filters to graph neural networks. *IEEE Signal Processing Magazine*, 37:128–138, November 2020. ISSN 1558-0792.

[57] Fernando Gama, Joan Bruna, and Alejandro Ribeiro. Stability properties of graph neural networks. *IEEE Transactions on Signal Processing*, 68:5680–5695, 2020.

[58] Samar Hadou, Charilaos I Kanatsoulis, and Alejandro Ribeiro. Space-time graph neural networks with stochastic graph perturbations. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023.

[59] Tao Sun, Dongsheng Li, and Bao Wang. Decentralized federated averaging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4289–4301, 2023.

[60] H. B. McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *International Conference on Artificial Intelligence and Statistics*, 2016.

[61] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, pages 5132–5143. PMLR, 2020.

[62] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10713–10722, June 2021.

[63] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

[64] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas Navarro, Matthew Mattina, Paul N Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. *arXiv preprint arXiv:2111.04263*, 2021.

[65] H. Robbins and D. Siegmund. A convergence theorem for non negative almost supermartingales and some applications. In *Optimizing Methods in Statistics*, pages 233–257. Academic Press, January 1971.

[66] Rick Durrett. *Probability: theory and examples*, volume 49. Cambridge university press, 2019.

[67] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

A. Constrained Learning Theory

In this section, we provide a rigorous statement for CLT theorem and the assumptions under which it holds.

Assumption 1. *The loss function $\widehat{f}(\cdot)$ in (3) and the gradient norm $\|\nabla \widehat{f}(\cdot)\|$ are both bounded and M-Lipschitz continuous functions.*

Assumption 2. *Let $\widehat{\mathbb{E}}$ be the sample mean evaluated over Q realizations. Then there exists $\zeta(Q,\delta) \geq 0$ that is monotonically decreasing with Q, for which it holds with probability $1-\delta$ that*

1. $|\mathbb{E}[\widehat{f}(\boldsymbol{\Phi}(\boldsymbol{\vartheta};\boldsymbol{\theta}))] - \widehat{\mathbb{E}}[\widehat{f}(\boldsymbol{\Phi}(\boldsymbol{\vartheta};\boldsymbol{\theta}))]| \leq \zeta(Q,\delta)$, *and*
2. $|\mathbb{E}[\|\nabla \widehat{f}(\mathbf{W}_l(\boldsymbol{\vartheta};\boldsymbol{\theta}))\|] - \widehat{\mathbb{E}}[\|\nabla \widehat{f}(\mathbf{W}_l(\boldsymbol{\vartheta};\boldsymbol{\theta}))\|]| \leq \zeta(Q,\delta)$ *for all l and all $\boldsymbol{\theta} \in \mathbb{R}^p$.*

Assumption 3. *Let $\phi_{l^\circ} \ldots \circ \phi_1 \in \mathcal{P}_l$ be a composition of l unrolled layers parameterized by $\boldsymbol{\theta}_{1:l}$ and $\overline{\mathcal{P}}_l = \overline{conv}(\mathcal{P}_l)$ be the convex hull of $\mathcal{P}_l$. Then, for each $\overline{\phi}_{l^\circ} \ldots \circ \overline{\phi}_1 \in \overline{\mathcal{P}}$ and $\nu > 0$, there exists $\boldsymbol{\theta}_{1:l}$ such that $\mathbb{E}\left[|\phi_{l^\circ} \ldots \circ \phi_1(\mathbf{W}_0, \boldsymbol{\vartheta}; \boldsymbol{\theta}_{1:l}) - \overline{\phi}_{l^\circ} \ldots \circ \overline{\phi}_1(\mathbf{W}_0, \boldsymbol{\vartheta})|\right] \leq \nu$ for all l.*

Assumption 4. *There exists $\boldsymbol{\Phi} \in \mathcal{H}$ that is strictly feasible, i.e., $\mathbb{E}\big[\|\nabla \widehat{f}(\mathbf{W}_l)\| - (1-\epsilon)\,\|\nabla \widehat{f}(\mathbf{W}_{l-1})\|\big] < -M\nu, \forall l$, with M and ν as in Assumptions 1 and 3.*

Assumption 5. *The conditional distribution $p(\boldsymbol{\vartheta}|\mathbf{W})$ is non-atomic for all $\mathbf{W}$.*

The above assumptions can be easily satisfied in practice. Assumption 1 requires the loss function and its gradient to be smooth and bounded. Assumption 2 identifies the sample complexity, which is a common assumption when handling statistical models. Moreover, Assumption 3 forces the parameterization $\boldsymbol{\theta}_l$ to be sufficiently rich at each layer l, which is guaranteed by modern deep learning models. Assumption 4 ensures that the problem is feasible and well posed, which is guaranteed since (3) mimics the parameters of a standard iterative solution. Finally, Assumption 5 can be satisfied using data augmentation.

Theorem 4 (CLT [54])**.** *Let P^* be the optimal value of (3) and $(\boldsymbol{\theta}^*, \boldsymbol{\lambda}^*)$ be a stationary point of (4). Under Assumptions 1- 5, it holds, for some constant ρ, that*

$$|P^* - \widehat{D}^*| \leq M\nu + \rho\,\zeta(Q,\delta), \text{ and} \tag{14}$$

$$\mathbb{E}\big[\|\nabla \widehat{f}(\mathbf{W}_l)\| - (1-\epsilon)\,\|\nabla \widehat{f}(\mathbf{W}_{l-1})\|\big] \leq \zeta(Q,\delta), \quad \forall l, \tag{15}$$

with probability $1-\delta$ each and with $\rho \geq \max\{\|\boldsymbol{\lambda}^*\|, \|\overline{\boldsymbol{\lambda}}^*\|\}$, where $\overline{\boldsymbol{\lambda}}^* = \operatorname{argmax}_{\boldsymbol{\lambda}} \min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda})$.

CLT asserts that the gap between the two problems is affected by a smoothness constant M, the richness of the parameterization $\boldsymbol{\theta}$, and the sample complexity.

B. Proofs

In this section, we provide the proofs for our theoretical results after introducing the following notation. Consider a probability space $(\Omega, \mathcal{F}, P)$, where Ω is a sample space, $\mathcal{F}$ is a sigma algebra, and $P : \mathcal{F} \to [0,1]$ is a probability measure. With a slight abuse of this measure-theoretic notation, we write $P(X=0)$ instead of $P(\{\omega : X(\omega) = 0\})$, where $X : \Omega \to \mathbb{R}$ is a random variable, to keep equations concise. We define a filtration of $\mathcal{F}$ as $\{\mathcal{F}_l\}_{l>0}$, which can be thought of as an increasing sequence of σ-algebras with $\mathcal{F}_{l-1} \subset \mathcal{F}_l$. We assume that the outputs of the unrolled layers $\mathbf{W}_l$ are adapted to $\mathcal{F}_l$, i.e., $\mathbf{W}_l \in \mathcal{F}_l$. Intuitively, the filtration $\mathcal{F}_l$ describes the information at our disposal at step l, which includes the outputs of each layer up to layer l, along with the initial estimate $\mathbf{W}_0$.

In our proofs, we use a supermartingale argument, which is commonly used to prove the convergence of stochastic descent algorithms. A stochastic process X_k is said to form a supermartingale if $\mathbb{E}[X_k|X_{k-1}, \ldots, X_0] \leq X_{k-1}$. This inequality implies that given the past history of the process, the future value X_k is not, on average, larger than the latest one. With this definition in mind, we provide the proof of Theorem 2.

B.1. Proof of Theorem 2

This proof follows the lines of the proof of Theorem 2 in [25].

Let $A_l \in \mathcal{F}_l$ be the event that the constraint (15) at layer l is satisfied. By the law of total expectation, we have

$$\mathbb{E}\big[\|\nabla \widehat{f}(\mathbf{W}_l)\|\ \big] = P(A_l)\mathbb{E}\Big[\|\nabla \widehat{f}(\mathbf{W}_l)\|\ |A_l\Big] + P(A_l^c)\mathbb{E}\Big[\|\nabla \widehat{f}(\mathbf{W}_l)\|\ |A_l^c\Big], \tag{16}$$

with $P(A_l) = 1 - \delta$. On the right-hand side, the first term represents the conditional expectation when the constraint is satisfied and, in turn, is bounded above according to (15). The second term is concerned with the complementary event $A_l^c \in \mathcal{F}_l$, when the constraint is violated. The conditional expectation in this case can also be bounded since i) the gradient norm $\|\nabla \widehat{f}(\mathbf{W}_l)\| \leq M$ for all $\mathbf{W}_l$ since f is M-Lipschitz according to Assumption 1, and ii) the expectation of a random variable cannot exceed its maximum value, i.e, $\mathbb{E}\|\nabla \widehat{f}(\mathbf{W}_l)\| \leq \max_{\mathbf{W}_l} \|\nabla \widehat{f}(\mathbf{W}_l)\| \leq M$ by Cauchy-Schwarz inequality. Substituting in (16) results in an upper bound of

$$\mathbb{E}\big[\|\nabla \widehat{f}(\mathbf{W}_l)\|\big] \leq (1-\delta)(1-\epsilon)\ \mathbb{E}\|\nabla \widehat{f}(\mathbf{W}_{l-1})\| + (1-\delta)\zeta(Q,\delta) + \delta M, \tag{17}$$

almost surely.

In the rest of the proof, we leverage the supermartingale convergence theorem to show that (17) indeed implies the required convergence. We start by defining a sequence of random variables $\{Z_l\}_l$ each of which has a degenerative distribution such that

$$Z_l = \mathbb{E}\|\nabla \widehat{f}(\mathbf{W}_l)\| \quad a.s. \quad \forall l. \tag{18}$$

Then, we form a supermartingale-like inequality using the law of total expectation. That is, we have

$$\begin{aligned} \mathbb{E}[Z_l|\ \mathcal{F}_{l-1}] &\leq (1-\delta)(1-\epsilon)\ Z_{l-1} + (1-\delta)\zeta(Q,\delta) + \delta M \\ &= (1-\delta)\ Z_{l-1} - (1-\delta)\Big(\epsilon Z_{l-1} - \zeta(Q,\delta) - \frac{\delta M}{1-\delta}\Big). \end{aligned} \tag{19}$$

The structure of the proof is then divided into two steps. First, we prove that when l grows, Z_l almost surely and infinitely often achieves values below $\frac{1}{\epsilon}\big(\zeta(Q,\delta) + \delta M/1-\delta\big)$. Second, we show that this is also true for the gradient norm $\|\nabla \widehat{f}(\mathbf{W}_l)\|$ itself. This implies that the outputs of the unrolled layers enter a near-optimal region infinitely often.

To tackle the first objective, we define the lowest gradient norm achieved, on average, up to layer l as $Z_l^{\text{best}} = \min_{k\leq l}\{Z_k\}$. To ensure that Z_l enters this region infinitely often, it suffices to show that

$$\lim_{l\to\infty} Z_l^{\text{best}} \leq \frac{1}{\epsilon}\big(\zeta(Q,\delta) + \delta M/1-\delta\big) \quad a.s. \tag{20}$$

To show that the above inequality is true, we start by defining the sequences

$$\begin{aligned} \alpha_l &:= Z_l \cdot \mathbf{1}\Big\{Z_l^{\text{best}} > \frac{1}{\epsilon}\big(\zeta(Q,\delta) + \delta M/1-\delta\big)\Big\}, \\ \beta_l &:= \Big(\epsilon Z_l - \zeta(Q,\delta) - \frac{\delta M}{1-\delta}\Big)\mathbf{1}\Big\{Z_l^{\text{best}} > \frac{1}{\epsilon}\big(\zeta(Q,\delta) + \delta M/1-\delta\big)\Big\}, \end{aligned} \tag{21}$$

where $\mathbf{1}\{.\}$ is an indicator function. The first sequence α_l tracks the values of Z_l until the best value Z_l^{best} drops below the threshold $\frac{1}{\epsilon}\big(\zeta(Q,\delta) + \delta M/1-\delta\big)$ for the first time. After this point, the best value stays below the threshold since $Z_{l+1}^{\text{best}} \leq Z_l^{\text{best}}$ by definition, which implies that the indicator function stays zero and $\alpha_l = 0$. In other words, we have

$$\alpha_l = \begin{cases} Z_l & l < T \\ 0 & \text{otherwise}, \end{cases} \tag{22}$$

with $T := \min\{l \mid Z_l^{\text{best}} \leq \frac{1}{\epsilon}\big(\zeta(Q,\delta) + \delta M/1-\delta\big)\}$. Similarly, the sequence β_l follows the values of $\epsilon Z_l - \zeta(Q,\delta) - \frac{\delta}{1-\delta}M$ until it falls below zero for the first time, which implies that $\beta_l \geq 0$ by construction. Moreover, it also holds that $\alpha_l \geq 0$ for all l since Z_l is always non-negative.

We now aim to show that α_l forms a supermartingale, so we can use the supermartingale convergence theorem to prove (20). This requires finding an upper bound of the conditional expectation $\mathbb{E}[\alpha_l|\mathcal{F}_{l-1}]$. We separate this expectation into two cases, $\alpha_{l-1}=0$ and $\alpha_{l-1}\neq 0$, and use the law of total expectation to write

$$\mathbb{E}[\alpha_l|\mathcal{F}_{l-1}] = \mathbb{E}[\alpha_l|\mathcal{F}_{l-1},\alpha_{l-1}=0]P(\alpha_{l-1}=0) + \mathbb{E}[\alpha_l|\mathcal{F}_{l-1},\alpha_{l-1}\neq 0]P(\alpha_{l-1}\neq 0). \tag{23}$$

First, we focus on the case when $\alpha_{l-1}=0$, and for conciseness, let $\eta := \frac{1}{\epsilon}\big(\zeta(Q,\delta)+\delta M/(1-\delta)\big)$ be the radius of the near-optimal region centered around the optimal. Equation (21) then implies that the indicator function is zero, i.e., $Z_l^{\text{best}}\leq\eta$, since the non-negative random variable Z_l cannot be zero without $Z_l^{\text{best}}\leq\eta$. It also follows that β_{l-1} is zero since it employs the same indicator function as α_l. As we discussed earlier, once $\alpha_{l-1}=0$, all the values that follow are also zero, i.e., $\alpha_k=0,\ \forall k\geq l-1$ (c.f. (22)). Hence, the conditional expectation of α_l can be written as

$$\mathbb{E}[\alpha_l|\mathcal{F}_{l-1},\alpha_{l-1}=0] = 0 =: (1-\delta)(\alpha_{l-1}-\beta_{l-1}). \tag{24}$$

On the other hand, when $\alpha_{l-1}\neq 0$, the conditional expectation follows from the definition in (21),

$$\begin{aligned}
\mathbb{E}[\alpha_l|\mathcal{F}_{l-1},\alpha_{l-1}\neq 0] &= \mathbb{E}[Z_l\cdot\mathbf{1}\{Z_l^{\text{best}}>\eta\}|\mathcal{F}_{l-1},\alpha_{l-1}\neq 0]\\
&\leq \mathbb{E}[Z_l|\mathcal{F}_{l-1},\alpha_{l-1}\neq 0]\\
&\leq (1-\delta)\ Z_{l-1} - (1-\delta)\Big(\epsilon Z_{l-1}-\zeta(Q,\delta)-\frac{\delta M}{1-\delta}\Big)\\
&= (1-\delta)(\alpha_{l-1}-\beta_{l-1}).
\end{aligned} \tag{25}$$

The first inequality is true since the indicator function is at most one, and the second inequality is a direct application of (19). The last equality results from the fact that the indicator function $\mathbf{1}\{Z_l^{\text{best}}>\eta\}$ is 1 since $\alpha_{l-1}\neq 0$, which implies that $\alpha_{l-1}=Z_{l-1}$ and $\beta_{l-1}=\epsilon Z_{l-1}-\zeta(Q,\delta)-\frac{\delta}{1-\delta}M$. Combining the results in (24) and (25) and substituting in (23), it finally follows that

$$\begin{aligned}
\mathbb{E}[\alpha_l|\mathcal{F}_{l-1}] &\leq (1-\delta)(\alpha_{l-1}-\beta_{l-1})[P(\alpha_{l-1}=0)+P(\alpha_{l-1}\neq 0)]\\
&= (1-\delta)(\alpha_{l-1}-\beta_{l-1}),
\end{aligned} \tag{26}$$

and we emphasize that both α_{l-1} and β_{l-1} are non-negative by definition.

Given (26), it follows from supermartingale convergence theorem [65, Theorem 1] that (i) α_l converges almost surely, and (ii) $\sum_{l=1}^{\infty}\beta_l$ is almost surely summable (i.e., finite). When the latter is written explicitly, we get

$$\sum_{l=1}^{\infty}\Big(\epsilon Z_l-\zeta(Q,\delta)-\frac{\delta M}{1-\delta}\Big)\mathbf{1}\{Z_l^{\text{best}}>\eta\}<\infty,\quad a.s., \tag{27}$$

The almost sure convergence of the above sequence implies that the limit inferior and limit superior coincide and

$$\liminf_{l\to\infty}\Big(\epsilon Z_l-\zeta(Q,\delta)-\frac{\delta M}{1-\delta}\Big)\mathbf{1}\{Z_l^{\text{best}}>\eta\}=0,\quad a.s. \tag{28}$$

The latter is true if either there exist a sufficiently large l such that $Z_l^{\text{best}}\leq\eta=\frac{1}{\epsilon}\big(\zeta(Q,\delta)+\delta M/1-\delta\big)$ or it holds that

$$\liminf_{l\to\infty}\Big(\epsilon Z_l-\zeta(Q,\delta)-\frac{\delta M}{1-\delta}\Big)=0,\quad a.s. \tag{29}$$

The above equation can be re-written as $\sup_l\inf_{m\geq l}Z_m=\frac{1}{\epsilon}\big(\zeta(Q,\delta)+\frac{\delta M}{1-\delta}\big)$. Hence, there exists some large l where $Z_l^{\text{best}}\leq\sup_l\inf_{m\geq l}Z_m$, which again reaches the same conclusion. This proves the correctness of (20).

To this end, we have shown the convergence of Z_l^{best}, which was defined as the best *expected* value of $\|\nabla\widehat{f}(\mathbf{W}_l)\|$. It is still left to show the convergence of the random variable $\|\nabla\widehat{f}(\mathbf{W}_l)\|$ itself. Start with writing $Z_l=\int\|\nabla\widehat{f}(\mathbf{W}_l)\|dP$, which turns (29) into

$$\liminf_{l\to\infty}\int\epsilon\|\nabla\widehat{f}(\mathbf{W}_l)\|dP=\zeta(Q,\delta)+\frac{\delta M}{1-\delta},\quad a.s. \tag{30}$$

By Fatou's lemma [66, Theorem 1.5.5], it follows that

$$\int \liminf_{l\to\infty} \epsilon \|\nabla \widehat{f}(\mathbf{W}_l)\| dP \leq \liminf_{l\to\infty} \int \epsilon \|\nabla \widehat{f}(\mathbf{W}_l)\| dP = \zeta(Q,\delta) + \frac{\delta M}{1-\delta}. \tag{31}$$

We can bound the left hand side from below by defining $f_l^{\text{best}} := \min_{k\leq l} \|\nabla \widehat{f}(\mathbf{W}_k)\|$ as the lowest gradient norm achieved up to layer l. By definition, $f_l^{\text{best}} \leq \liminf_{l\to\infty} \|\nabla \widehat{f}(\mathbf{W}_l)\|$ for sufficiently large l. Therefore, we get

$$\epsilon \int f_l^{\text{best}} dP \leq \epsilon \int \liminf_{l\to\infty} \|\nabla \widehat{f}(\mathbf{W}_l)\| dP \leq \zeta(Q,\delta) + \frac{\delta M}{1-\delta}, \quad a.s. \tag{32}$$

for some large l. Equivalently, we can write that

$$\lim_{l\to\infty} \int f_l^{\text{best}} dP \leq \frac{1}{\epsilon}\left(\zeta(Q,\delta) + \frac{\delta M}{1-\delta}\right), \quad a.s. \tag{33}$$

which completes the proof.

B.2. Proofs of Lemma 1 and Theorem 3

Assumption 6. *There exists a non-negative symmetric distance d between the input distribution $\mathcal{D}_\mathcal{T}$ and the OOD distribution $\mathcal{D}'_\mathcal{T}$ such that*

$$\left| \mathbb{E}_{\mathcal{D}_\mathcal{T}}\left[\left\|\nabla \widehat{f}(\mathbf{W};\, \mathbf{B})\right\|\right] - \mathbb{E}_{\mathcal{D}'_\mathcal{T}}\left[\left\|\nabla \widehat{f}(\mathbf{W};\, \mathbf{B})\right\|\right] \right| \leq M d(\mathcal{D}_\mathcal{T}, \mathcal{D}'_\mathcal{T})$$

uniformly over $\mathbf{y}$ *with* M *being a Lipschitz constant.*

Proof. We start by adding and subtracting the following two quantities $\mathbb{E}_{\mathcal{D}_\mathcal{T}}\left[\left\|\nabla \widehat{f}(\mathbf{W}_l;\, \mathbf{B}_l)\right\|\right]$ and $(1-\epsilon)\mathbb{E}_{\mathcal{D}_\mathcal{T}}\left[\left\|\nabla \widehat{f}(\mathbf{W}_{l-1};\, \mathbf{B}_{l-1})\right\|\right]$, i.e., we get

$$\begin{aligned}
&\mathbb{E}_{\mathcal{D}'_\mathcal{T}}\left[\left\|\nabla \widehat{f}(\mathbf{W}_l;\, \mathbf{B}_l)\right\|\right] - (1-\epsilon)\, \mathbb{E}_{\mathcal{D}'_\mathcal{T}}\left[\left\|\nabla \widehat{f}(\mathbf{W}_{l-1};\, \mathbf{B}_{l-1})\right\|\right] \\
&= \mathbb{E}_{\mathcal{D}'_\mathcal{T}}\left[\left\|\nabla \widehat{f}(\mathbf{W}_l;\, \mathbf{B}_l)\right\|\right] - \mathbb{E}_{\mathcal{D}_\mathcal{T}}\left[\left\|\nabla \widehat{f}(\mathbf{W}_l;\, \mathbf{B}_l)\right\|\right] \\
&+ (1-\epsilon)\left[\mathbb{E}_{\mathcal{D}_\mathcal{T}}\left[\left\|\nabla \widehat{f}(\mathbf{W}_{l-1};\, \mathbf{B}_{l-1})\right\|\right] - \mathbb{E}_{\mathcal{D}'_\mathcal{T}}\left[\left\|\nabla \widehat{f}(\mathbf{W}_{l-1};\, \mathbf{B}_{l-1})\right\|\right]\right] \\
&+ \mathbb{E}_{\mathcal{D}_\mathcal{T}}\left[\left\|\nabla \widehat{f}(\mathbf{W}_l;\, \mathbf{B}_l)\right\|\right] - (1-\epsilon)\, \mathbb{E}_{\mathcal{D}_\mathcal{T}}\left[\left\|\nabla \widehat{f}(\mathbf{W}_{l-1};\, \mathbf{B}_{l-1})\right\|\right].
\end{aligned} \tag{34}$$

The right-hand side consists of three terms that can be bounded above with positive quantities according to Assumption 6 and (15). Therefore, the descent constraints under the new distribution $D_{x'}$ can be bounded above by

$$\begin{aligned}
&\mathbb{E}_{\mathcal{D}'_\mathcal{T}}\left[\left\|\nabla \widehat{f}(\mathbf{W}_l;\, \mathbf{B}_l)\right\|\right] - (1-\epsilon)\, \mathbb{E}_{\mathcal{D}'_\mathcal{T}}\left[\left\|\nabla \widehat{f}(\mathbf{W}_{l-1};\, \mathbf{B}_{l-1})\right\|\right] \\
&\quad \leq M d(\mathcal{D}_\mathcal{T}, \mathcal{D}'_\mathcal{T}) + M(1-\epsilon) d(\mathcal{D}_\mathcal{T}, \mathcal{D}'_\mathcal{T}) + \zeta(N,\delta) \\
&\quad \leq 2M d(\mathcal{D}_\mathcal{T}, \mathcal{D}'_\mathcal{T}) + \zeta(N,\delta).
\end{aligned} \tag{35}$$

Notice that this inequality holds with probability $1-\delta$ since the upper bound in (15) also holds with the same probability. This completes the proof. □

The proof of Theorem 3 follows directly from Theorem 2 and Lemma 1.

C. Experiments Details

Training. At each epoch, we randomly choose one image dataset from the meta-training dataset and feed its 45 training examples/agent to a 10-layer U-DGD network in mini-batches of 10 examples/agent at each layer). The training loss is computed over the 10 testing examples/agent and

optimized using ADAM with a learning rate $\mu_\theta = 10^{-2}$ and a dual learning rate $\mu_\lambda = 10^{-2}$. We utilize ReLU activation functions at each layer, and the constraint parameter ϵ is set to 0.01. The performance of the trained U-DGD is examined over a meta-testing dataset that consists of 30 class imbalanced datasets, each of which also has 45 training examples and 15 for testing per agent. Similar to training, the training examples are fed to the U-DGD in mini-batches while the testing examples are used to compute the testing accuracy. The results are reported for CIFAR10 dataset [67]. All experiments were run on an NVIDIA® GeForce RTX™ 3090 GPU.

Decentralized FL benchmarks. In both DGD and DSGD, the agents update their estimates based on their local data through one gradient step at each communication round. The gradients in DGD are computed over a mini-batch of 10 data points/agent compared to one data point in DSGD. In DFedAvgM, each agent takes 6 gradient steps with momentum at each communication round. The step sizes are 10^3, 10^4, 10^2 in DGD, DSGD, and DFedAvgM, respectively.