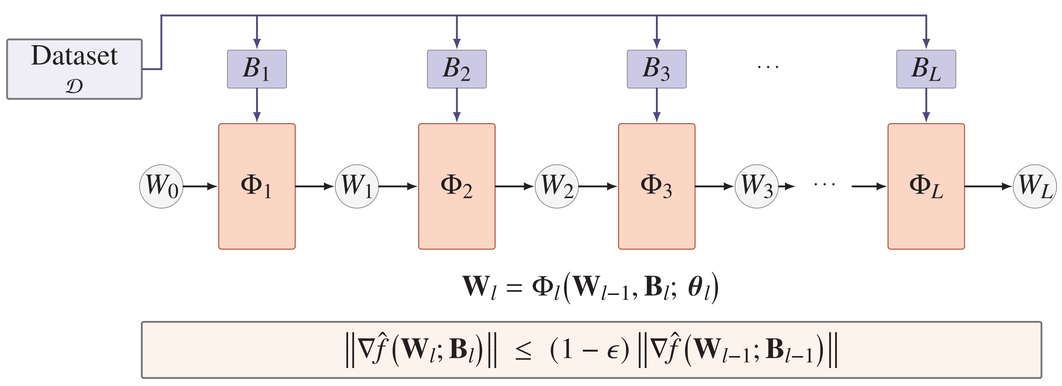
Stochastic Unrolled Neural Networks

Samar Hadou*, Navid NaderiAlizadeh† and Alejandro Ribeiro*

*University of Pennsylvania, †Duke University

Summary

- **Goal**: Train unrolled neural networks to solve empirical risk minimization (ERM) tasks.
  ⇒ **Advantage**: Faster convergence.
- **Challenge**: An ERM task is determined by a dataset that needs to be fed to the neural network as an input.
- **Our Solution**: We let each layer interact with randomly drawn mini-batches from the downstream dataset.
- **Features**: We impose descent constraints on the loss across the unrolled layers:
  ⇒ To achieve out-of-distribution (OOD) robustness.



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

$$\left\|\nabla \widehat{f}(\mathbf{W}_l; \mathbf{B}_l)\right\| \;\leq\; (1-\epsilon)\left\|\nabla \widehat{f}(\mathbf{W}_{l-1}; \mathbf{B}_{l-1})\right\|$$

Stochastic Unrolling

- An ERM task is to find a neural network parameterization $\mathbf{W}$ that minimizes a loss $\widehat{f}$,

$$\mathbf{W}^*_{task} \;=\; \operatorname*{argmin}_{\mathbf{W}} \; \widehat{f}\big(\boldsymbol{\Phi}(\mathcal{D}; \mathbf{W});\, \mathcal{D}\big).$$

- Consider **a family of tasks** for which we train an unrolled network as a learned optimizer:

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

- Unrolling aims to learn the parameters in a conventional iterative rule Φ_l (e.g. GD):

$$\mathbf{W}_l = \Phi_l\big(\mathbf{W}_{l-1}, \mathcal{D}\,; \boldsymbol{\theta}_l\big), \quad \forall l.$$

⇒ Challenge: Datasets are too huge to be fed to conventional neural networks.

- Our solution is to feed each layer an independent mini-batch drawn i.i.d. from the dataset,

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

- Stochastic Unrolling vs Meta-learning:
  ⇒ *Meta-learning* → $\mathbf{W}_{meta}$ that solves any task directly with little to no adaptation.
  ⇒ *Unrolling* → optimizer $\boldsymbol{\Theta}$ → a parameterization $\mathbf{W}_{task}$ for each downstream task.

Training with Descent Constraints

- We also enforce **descent constraints** across the unrolled layers such that the network mimics a stochastic descent algorithm,

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

- To solve this constrained learning problem, we construct the Lagrangian function as

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

- We then execute a primal-dual algorithm to find the saddle point:

$$\boldsymbol{\Theta} = \boldsymbol{\Theta} - \eta_1 \nabla_{\boldsymbol{\Theta}} \widehat{\mathcal{L}}(\boldsymbol{\Theta}, \boldsymbol{\lambda}), \qquad \boldsymbol{\lambda} = \Big[\boldsymbol{\lambda} + \eta_2 \nabla_{\boldsymbol{\lambda}} \widehat{\mathcal{L}}(\boldsymbol{\Theta}, \boldsymbol{\lambda})\Big]_+.$$

Convergence (informal)

Assume that the loss function $\widehat{f}(\mathbf{W})$ is M-Lipschitz. Then, it holds that

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

where $\zeta(N, \delta)$ is the sample complexity term.

OOD Generalization (informal)

Training on a task distribution $\mathcal{D}_{\mathcal{T}}$ and evaluating on $\mathcal{D}'_{\mathcal{T}}$ yields

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

where $d(\cdot, \cdot)$ is a distance metric between the two distributions.

Stochastic Unrolled Federated Learning (SURF)

- Consider a federated setting, where n agents collaborate to train a neural network $\mathbf{w}$:

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

⇒ $\mathbf{w}_i$ is the parameters learned by agent i and we seek consensus among the agents.

- The decentralized gradient descent (DGD) generates a sequence of iterates of the form:

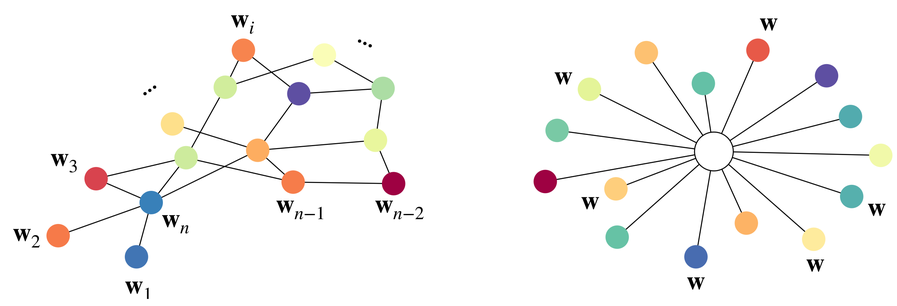
$$\mathbf{w}_i(l) \;=\; \sum_{j\in\mathcal{N}_i\cup\{i\}} \alpha_{ij}\mathbf{w}_j(l-1) - \beta\nabla \widehat{f}_i(\mathbf{w}_i(l-1)), \;\forall i.$$

⇒ A linear aggregation of direct neighbors followed by a *local* gradient update.

- DGD iteration is unfolded into a graph filter (GF) and a *local* fully-connected network,

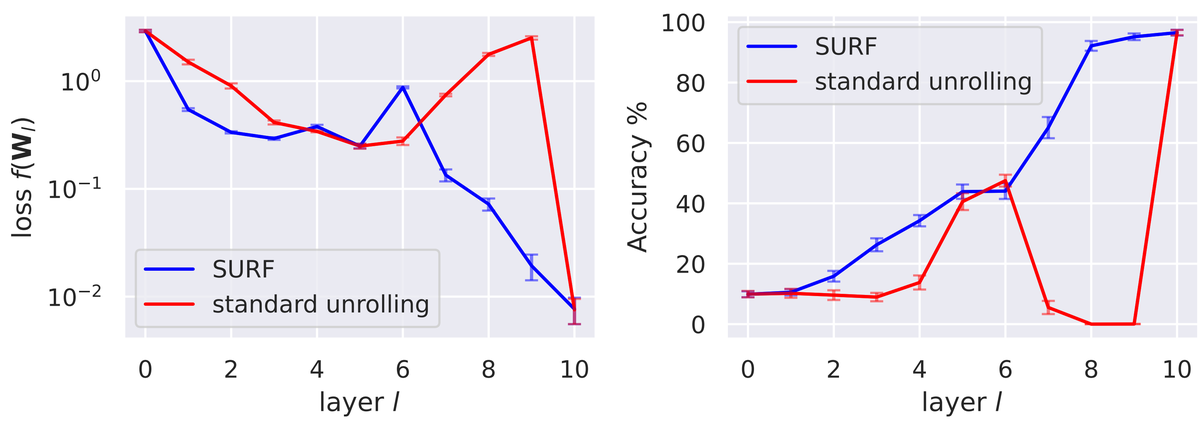
$$\mathbf{w}_{i,l} \;=\; \big[\mathbf{H}_l(\mathbf{W}_{l-1})\big]_i - \sigma\Big(\mathbf{M}_l\big[\mathbf{w}_{i,l-1} \| \mathbf{b}_{i,l}\big] + \mathbf{d}_l\Big).$$

- The GF aggregates information from up to K-hop neighbors: $\mathbf{H}(\mathbf{W}_{l-1}) = \sum_{k=0}^{K} h_{k,l}\mathbf{S}^k\mathbf{W}_{l-1}$.



Numerical Results

- **Task**: A network of $n = 100$ agents trains the softmax layer of an image classifier.
- Convergence and ablation of constraints:
  ⇒ Imposing descent constraints preserves monotone improvement across the layers.

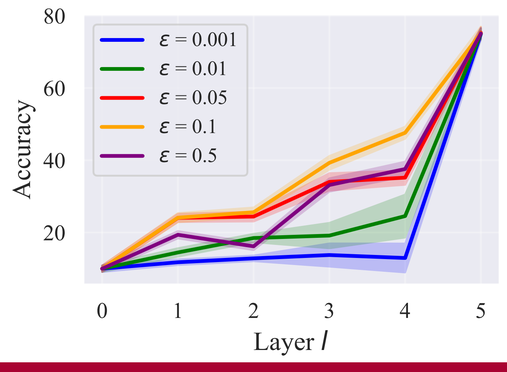


- **Role of constraint parameter** ϵ: All values yield the same last-layer performance.

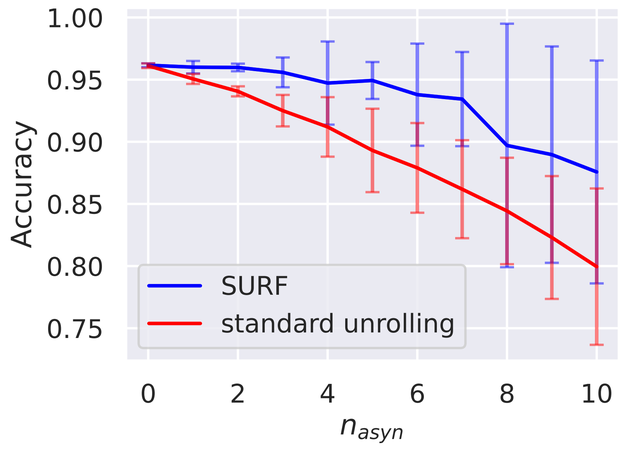
⇒ **Small values** → trivial constraints → no monotone progress (cf. wider convergence regions).

⇒ **Increasing** ϵ → progressively faster convergence.

⇒ **Large values** → harder-to-satisfy constraints (more violations).



- OOD generalization under **asynchronous communications**.
  ⇒ At each layer, n_{asyn} **random agents** do not update or send their gradients.
  ⇒ This induces **distribution shifts** at the input of each layer.
  ⇒ The constrained model generalizes better under this distribution shift.



References:

[1] S. Hadou, N. NaderiAlizadeh, and A. Ribeiro, "Robust stochastically-descending unrolled networks," IEEE Transactions on Signal Processing, vol. 72, pp. 5484-5499, 2024.

[2] L. F. Chamon, S. Paternain, M. Calvo-Fullana, and A. Ribeiro, "Constrained learning with non-convex losses," IEEE Transactions on Information Theory, 2022.

[3] Samar Hadou and Alejandro Ribeiro. Unrolled neural networks for constrained optimization, 2026. URL https://arxiv.org/abs/2601.17274.