OPT2023: 15th Annual Workshop on Optimization for Machine Learning

An alternative approach to train neural networks using monotone variational inequality

Chen Xu[1] CXU310@GATECH.EDU
Xiuyuan Cheng[2] XIUYUAN.CHENG@DUKE.EDU
Yao Xie[1] YAO.XIE@ISYE.GATECH.EDU
[1]Georgia Institute of Technology, [2]Duke University

Abstract

We investigate an alternative approach to neural network training, which is a non-convex optimization problem, through the lens of another convex problem — to solve a monotone variational inequality (MVI) - inspired by the work of [13]. MVI solutions can be found by computationally efficient procedures, with performance guarantee of ℓ_2 and ℓ_∞ bounds on model recovery and prediction accuracy under the theoretical setting of training a single-layer linear neural network. We study the use of MVI for training multi-layer neural networks by proposing a practical and completely general algorithm called *stochastic variational inequality* (`SVI`). We demonstrate its applicability in training networks with various architectures (`SVI` is completely general for training any network). We show the competitive or better performance of `SVI` compared to the widely-used stochastic gradient descent method (SGD) on both synthetic and real data prediction tasks regarding various performance metrics, especially in the improved efficiency in the early stage of training.

1. Introduction

Neural network (NN) training [12, 15, 26, 27] is the essential process in the study of deep models. Optimization guarantee for training loss as well as generalization error have been obtained with over-parametrized networks [1–3, 6, 20, 21]. However, due to the inherent non-convexity of loss objectives, theoretical developments are still diffused and lag behind the vast empirical successes.

Recently, the seminal work [13] presented a somewhat surprising result: that some non-convex issues can be circumvented in special cases through problem reformulation. In particular, it was shown that one can formulate the parameter estimation problem of generalized linear models (GLM) as solving a monotone variational inequality (MVI), a general form of convex optimization. This differs from minimizing a least-square loss function, which leads to a non-convex optimization problem, and thus, no guarantees can be obtained for global convergence or model recovery. Consequently, the formulation through MVI leads to strong performance guarantees and computationally efficient procedures.

In this paper, drawing inspiration from [13] and considering the fact that certain GLM (such as logistic regression) can be viewed as the simplest network with only one layer, we propose a new scheme for training generic neural networks based on MVI. This marks a significant departure from the widely employed gradient descent algorithm for neural network training. In our approach, we replace the gradient of a loss function with operators inspired by MVI theory. The advantages of this approach are as follows: (i) In special cases, we can establish strong training and prediction guarantees; (ii) For general cases, through extensive numerical studies on synthetic and real data,

our approach showcases faster convergence to a local solution compared to gradient descent in a comparable setup.

Related work. Variational inequality has been studied mainly in the optimization problem context [7, 14]. More recently, MVI has been used to solve min-max problems in Generative Adversarial Networks [19] and reinforcement learning [16]. Our theory and techniques are inspired by [13], but we offer a thorough investigation of using MVI to train multi-layer neural networks. A recent work [4] proposed a similar VI-inspired approach in training neural networks, with corresponding convergence guarantees of model parameters. In contrast to our `SVI`, their proposed method cannot train multiple layers simultaneously and the method only demonstrated improved empirical performance over gradient-based methods on training the last layer of deep neural network.

Our contribution. First, we develop a general and practical algorithm called *stochastic variational inequality* (`SVI`) to train multi-layer neural networks. Second, we reformulate the last-layer neural network training as solving a monotone variational inequality, with guarantees on model recovery and prediction accuracy. Third, we compare `SVI` with widely-used stochastic gradient descent methods to demonstrate that `SVI` is flexible and competitive on various tasks, especially yielding faster convergence in the early stage of training.

MVI preliminaries. Given a parameter set $\boldsymbol{\theta} \subset \mathbb{R}^p$, we call a continuous mapping (operator) $F : \boldsymbol{\theta} \to \mathbb{R}^p$ *monotone* if for all $\theta_1, \theta_2 \in \boldsymbol{\theta}$, $\langle F(\theta_1) - F(\theta_2), \theta_1 - \theta_2 \rangle \geq 0$ [13]. The operator is called *strongly monotone* with modulus κ if for all $\theta_1, \theta_2 \in \boldsymbol{\theta}$,

$$\langle F(\theta_1) - F(\theta_2), \theta_1 - \theta_2 \rangle \geq \kappa \|\theta_1 - \theta_2\|_2^2. \tag{1}$$

For a monotone operator F on $\boldsymbol{\theta}$, the problem VI$[F, \boldsymbol{\theta}]$ is to find an $\bar{\theta} \in \boldsymbol{\theta}$ such that for all $\theta \in \boldsymbol{\theta}$,

$$\langle F(\bar{\theta}), \theta - \bar{\theta} \rangle \geq 0. \qquad \text{VI}[F, \boldsymbol{\theta}] \tag{2}$$

It is known that if $\boldsymbol{\theta}$ is compact, then (2) has at least one solution [22, Proposition 4.1]. In addition, if $\kappa > 0$ in (1), then (2) has exactly one solution [22, Proposition 4.4]. Under mild computability assumptions, the solution can be efficiently solved to high accuracy using iterative schemes [13].

2. MVI for neural network training

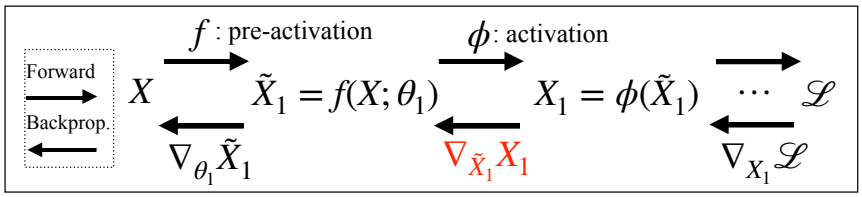
Assume a generic feature $X \in \mathbb{R}^d$, where d denotes the feature dimension. Suppose the conditional expectation $\mathbb{E}[Y|X]$ of the categorical[1] response vector $Y \in \{1, \ldots, K\}$ is modeled by an L-layer neural network $G(X, \theta)$:

$$\mathbb{E}[Y|X, \theta] = G(X, \theta) = \phi_L(g_L(X_L, \theta_L)), \tag{3}$$

where $X_L = \phi_{L-1}(g_{L-1}(X_{L-1}, \theta_{L-1})), X_1 = X$ denote the nonlinear feature transformation from the previous layer, $\theta = \{\theta_1, \ldots, \theta_L\}$ denotes model parameters, and each ϕ_l denotes the non-decreasing activation function at layer l. In particular, assume there exists θ^* so that $\mathbb{E}[Y|X] = G(X, \theta^*)$.

Our goal is to train θ so as to approximate $\mathbb{E}[Y|X]$ as closely as possible.

[1]The techniques and theory in this work can be used to model the conditional expectation of continuous random variables.



$$\Delta^{GD}\theta_1 = \nabla_{X_1}\mathscr{L} \circ \nabla_{\tilde{X}_1}X_1 \circ \nabla_{\theta_1}\tilde{X}_1 \quad \text{(Backprop.)}$$
$$\Delta^{SVI}\theta_1 = \nabla_{X_1}\mathscr{L} \circ \nabla_{\theta_1}\tilde{X}_1 \quad \text{(Backprop. with skipping)}$$

Figure 1: Gradient descent (GD) vs. `SVI`: the only difference lies in the skipping of differentiation of the activation ϕ with respect to pre-activation values.

Single-layer training. Suppose $L = 1$ in (3) with a particular form of pre-activation: $g(X, \theta) = \eta(X)\theta$, where $\eta(X)$ is a fixed feature transformation. Such network formulation is typically the last layer of a deep neural network, and it is mathematically equivalent to a GLM [13]. As in [13], we construct the monotone operator F as

$$F(\theta) = \mathbb{E}_{X,Y}\{\eta^\intercal(X)[\phi(\eta(X)\theta) - Y]\}, \tag{4}$$

where $\eta^\intercal(X)$ denotes the transpose of $\eta(X)$. Denote $\hat{F}$ as the empirical sample version of F. Then the typical training of θ based on MVI is the projected gradient descent

$$\theta \leftarrow \text{Proj}_{\boldsymbol{\theta}}(\theta - \gamma\hat{F}(\theta)). \tag{5}$$

where $\gamma > 0$ is the step-size and $\text{Proj}_{\boldsymbol{\theta}}$ projects back estimates to the feasible parameter domain $\boldsymbol{\theta}$. Note that (5) differs from the vanilla stochastic gradient descent (SGD) where the gradient of a certain loss objective takes the role of the monotone operator $\hat{F}$.

Multi-layer training. We first present a mathematically equivalent view of $F(\theta)$ in (4) for single-layer training. Assume we have the mean-squared-error (MSE) loss $\mathcal{L}(\hat{Y}, Y) = 1/2\|\hat{Y} - Y\|_2^2$ and the pre-activation mapping $\tilde{X} = \eta(X)\theta$. Denote $\hat{Y} = \phi(\tilde{X})$ as the prediction. By chain rule

$$F(\theta) := \mathbb{E}_{X,Y}\{\eta^\intercal(X)[\phi(\eta(X)\theta) - Y]\} = \mathbb{E}_{X,Y}\{\nabla_{\hat{Y}}\mathcal{L} \circ \nabla_\theta\tilde{X}\}. \tag{6}$$

Note that (6) comprises of two terms. The first term $\nabla_{\hat{Y}}\mathcal{L}$ is the gradient of the loss objective with respect to the network prediction $\hat{Y}$. The second term $\nabla_\theta\tilde{X}$ is the gradient of the *pre-activation* mapping $\tilde{X}$ with respect to the parameter θ. In particular, (6) is well-defined for any loss objective $\mathcal{L}$ and at any hidden layer l—$\tilde{X}$ would be the pre-activation value at the l-th layer, where $\hat{Y}$ would be the post-activation value at the l-th layer.

Thus, we propose `SVI` in Algorithm 1 to train any neural network represented in (3); Figure 1 illustrates the idea. In retrospect, compared to the commonly used gradient $\nabla_\theta\mathcal{L}$ in backpropagation, Algorithm 1 only differs by *skipping* the derivative of the point-wise non-linearity ϕ with respect to its input. Therefore, `SVI` barely differs in terms of computational cost against SGD.

There are two main benefits of Algorithm 1. First, the Algorithm is applicable to arbitrary form of network layers $g_l(X_l, \theta_l)$, nonlinear activations ϕ_l, and the loss function $\mathcal{L}$. Second, it is easy to implement by leveraging automatic differentiation [23]. Specifically, we implement the skipping idea

Algorithm 1 Stochastic variational inequality (SVI).

Require: Inputs (a) Training data $\{(X_i, Y_i)\}_{i=1}^N$, (b) An L-layer network $G(X, \theta) = \{\phi_l(g_l(X_l, \theta_l))\}_{l=1}^L$ as in (3), (c) Loss function $\mathcal{L} = \mathcal{L}(G(X, \theta), Y)$, (d) Learning rate $\gamma > 0$.
Ensure: Trained model parameters $\hat{\theta} = \{\hat{\theta}_l\}_{l=1}^L$

1: Initialize the network with some $\hat{\theta}$.
2: **while** Training **do**
3: Store $\{(\tilde{X}_{l+1}, X_{l+1})\}_{l=1}^L$ in forward pass on data, where $\tilde{X}_{l+1} = g_l(X_l, \hat{\theta}_l)$ and $X_{l+1} = \phi_l(\tilde{X}_{l+1})$.
4: Obtain $\{\nabla_{X_{l+1}} \mathcal{L}\}_{l=1}^L$.
5: **for** Layer $l = 1, \ldots, L$ **do**
6: Compute the surrogate loss $\tilde{\mathcal{L}}_l = \text{sum}(\tilde{X}_{l+1} \odot \nabla_{X_{l+1}} \mathcal{L})$.
7: Obtain $F_l(\hat{\theta}_l) = \nabla_{\hat{\theta}_l} \tilde{\mathcal{L}}_l$.
8: **end for**
9: Update $\hat{\theta}_l = \hat{\theta}_l - \eta F_l(\hat{\theta}_l)$ for $l = 1, \ldots, L$.
10: **end while**

via backpropagating the layer-wise surrogate loss $\tilde{\mathcal{L}}_l$ in line 6 with respect to θ_l. Note that this loss $\tilde{\mathcal{L}}_l$ is simple to compute: the quantity $\tilde{X}_{l+1}$ is available during the forward pass on training data X, and gradients $\nabla_{X_{l+1}} \mathcal{L}$ are available upon backpropagating the original loss $\mathcal{L}$ with respect to outputs of each layer l. In practice, one can also expect that SVI results in more significant amount of weight updates than SGD due to skipping (e.g., in the context of ReLU activation), which experimentally seems to speed up the initial model convergence. We illustrate this phenomenon in Figure 2.

3. Guarantee of model recovery by MVI

We now present training and estimation guarantees on model recovery for the *last-layer training*, assuming $g_L(X_L, \theta_L) = \eta_L(X_L)\theta_L$ in (3) and previous layers are estimated with ϵ accuracy in ℓ_2 norm. Specifically, let $\hat{\theta}_L^{(T)}$ be the estimated parameter after T training iterations. For a test feature X_t and the output $\hat{X}_{t,L}$, consider

$$\hat{\mathbb{E}}[Y_t|X_t] = \phi_L(\eta(\hat{X}_{t,L})\hat{\theta}_L^{(T)}), \tag{7}$$

which is the estimated prediction using $\hat{\theta}_L^{(T)}$. We will measure $\hat{\mathbb{E}}[Y_t|X_t]$ against the true model $\mathbb{E}[Y_t|X_t]$. More precisely, we will provide error bound on $\|\hat{\mathbb{E}}[Y_t|X_t] - \mathbb{E}[Y_t|X_t]\|_p, p \geq 2$ which crucially depends on the strong monotonicity modulus κ in (1) for $F(\theta_L)$.

Modulus κ greater than 0 We first state several assumptions used in convergence analyses.

Assumption 1 (ϵ-approximate estimate from previous layers) *Let $X_L^* = f_{1:L-1}^*(X)$ and $\hat{X}_L = \hat{f}_{1:L-1}(X)$ be the oracle and estimated output from the previous $L-1$ layers given a generic input feature X. There exists $\epsilon > 0$ such that $\mathbb{E}_X \|X_L^* - \hat{X}_L\|_2 \leq \epsilon$.*

Assumption 2 (Regularity conditions) *(1) The oracle parameter θ_L^* of the last layer satisfies the MVI inequality (2) for F. (2) The mapping $\eta(\cdot)$ is D-Lipschitz continuous. (3) For any $\theta_L \in \boldsymbol{\theta}$, $\|\theta_L\|_2 \leq B$.*

Proposition 3 (Prediction error bound for model recovery, strongly monotone F) *Under Assumptions 1 and 2, we have for a given test signal $X_t, t > N$ that for $p \in [2, \infty]$,*

$$\mathbb{E}_{X,\hat{\theta}_L^{(T)}}\{\|\hat{\mathbb{E}}[Y_t|X_t] - \mathbb{E}[Y_t|X_t]\|_p\} \leq (T+1)^{-1}C_t + C\epsilon,$$

where $\hat{\mathbb{E}}[Y_t|X_t]$ is defined in (7) under constants $C_t = [4M^2K\lambda_{\max}(\eta^\intercal(\hat{X}_{t,L})\eta(\hat{X}_{t,L}))]/\kappa^2$ and C=KDB. In particular, $p = 2$ yields the sum of squared error bound on prediction and $p = \infty$ yields entry-wise bound.

The same order of convergence holds when F in (4) is estimated by the empirical average of *mini-batches* of training data. In particular, the proof of Proposition 3 only requires access to an unbiased estimator of F so that the batch size can range from one to N, where N is the size of training data.

Modulus κ equal to 0 We may also encounter cases where the operator F is only monotone but not strongly monotone. For instance, let ϕ be the softmax function, which satisfies $\nabla\phi(z)\mathbf{1} = \mathbf{0}$ for any $z \in \mathbb{R}^n$ [9, Proposition 2]. Then, the minimum eigenvalue of $\nabla\phi(z)$ is always zero, leading to $\kappa = 0$. In this case, we use the extrapolation methoed (OE) [16] to obtain a similar but weaker ℓ_p prediction guarantee.

Proposition 4 (Prediction error bound for model recovery, monotone F) *Suppose we run the OE algorithm [16] for T iterations with $\lambda_t = 1$, $\gamma_t = [4K_2]^{-1}$, where K_2 is the Lipschitz constant of F. Let R be uniformly chosen from $\{2, 3, \ldots, T\}$. Then for $p \in [2, \infty]$,*

$$\mathbb{E}_{\hat{\theta}_L^{(R)}}\{\|\mathbb{E}_X\{\sigma_{\min}(\eta^\intercal(\hat{X}_L))[\hat{\mathbb{E}}[Y_t|X_t] - \mathbb{E}[Y_t|X_t]]\}\|_p\} \leq T^{-1/2}C_t'',$$

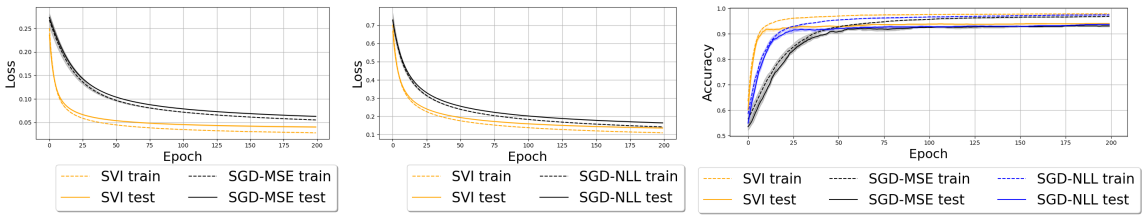
where $\sigma_{\min}(\cdot)$ denotes the minimum singular value of its input matrix and the constant $C_t'' = 3\sigma + 12K_2\sqrt{2\|\theta_L^*\|_2^2 + 2\sigma^2/L^2}$, in which $\sigma^2 = \mathbb{E}[(F_i(\theta_L) - F(\theta_L))^2]$ is the variance of the unbiased estimator.

The convergence rate in Proposition 4 is also unaffected by the batch size, which only serves to reduce the variance. In addition, Proposition 4 requires R be uniformly chosen from $\{2, 3, \ldots, T\}$, so that the theoretical guarantee holds at a random training epoch. In theory, this assumption is necessary to ensure a decrease of the norm of the monotone operator ([16], Eq. (3.20)). In practice, we observed that the epoch that leads to the highest validation accuracy might not occur at the end of T training epochs, so this assumption is reasonable based on empirical evidence.

4. Experiments

We test and compare `SVI` in Algorithm 1 with SGD on various synthetic and real-data examples. We demonstrate the consistently competitive or better performance by `SVI`, especially in the improved efficiency in the early stage of training against the SGD baseline. Additional details including hyperparameter selection are described in Appendix B.1.

Simulation on one-layer probit model We start with binary classification on data generated from a one-layer probit model, where the convergence of `SVI` model recovery is proven in Section 3. We generate data as follows: for each $i \geq 1$, $y_i \in \{0, 1\}$ and $\mathbb{E}[y_i|X_i, \theta^*] = \Phi(X_i^T\beta^* + b^*)$, where $X_i \in \mathbb{R}^p$, $X_{ij} \overset{i.i.d.}{\sim} \mathcal{N}(0.05, 1)$, $\beta_j^* \overset{i.i.d.}{\sim} \mathcal{N}(-0.05, 1)$, and $b^* \sim \mathcal{N}(-0.1, 1)$. Here, $\Phi(z) = \mathbb{P}(Z \leq$



(a) Losses under MSE objective (b) Losses under NLL objective (c) Accuracies on the same test data

Figure 2: One-layer FC model training. In (a) and (b), we plot training and test losses for both SGD (black) and `SVI` (orange). The loss choice for SGD is the mean-squared error (MSE) in (a) and the negative log-likelihood (NLL) in (b). The `SVI` update for one-layer training is based on (4), which does not depend on the loss objective, so we compute the MSE or NLL losses for comparison. In (c), we show the prediction accuracies by `SVI` in orange, SGD by MSE (SGD-MSE) in black, and SGD by NLL (SGD-NLL) in blue.

z) for $Z \sim \mathcal{N}(0, 1)$. Figure 2 shows the training and test performances by both methods. We see that `SVI` consistently yields smaller losses and higher prediction accuracies than SGD under both MSE and NLL objectives during training.

Real experiment on large graphs We demonstrate the applicability of `SVI` on the `ogbn-arxiv` graph provided by the Open Graph Benchmark [10, 11]. The graph nodes are arxiv papers to be classified into categories and edges denote citation among papers; this large graph has ∼170 thousand nodes, 1.16 million edges, 128-dimensional node features, and 40 node classes. We train four-layer GCN models of various hidden neuron sizes. Table 1 compares `SVI` against SGD under various network sizes. We see that `SVI` consistently reaches higher initial and final classification accuracies on training, validation, and test data.

Additional experiments In Appendix B, we perform additional simulation and real-data experiments. For simulation, we train GCN on graphs with increasing number of nodes (Appendix B.2). For real-data experiments, we consider solar ramping event detection (Appendix B.3), traffic flow anomaly detection (Appendix B.4), and image classification (Appendix B.5). In all experiments, we consistently observe faster initial convergence by `SVI` agianst SGD, with competitive or better final performance by `SVI` as well.

Table 1: Classification accuracies on the large `ogbn-arxiv` dataset under varying sizes of the GCN. "Initial" (resp. "Final") results indicate prediction accuracies after training 100 (resp. 1000) epochs. Entries in bracket show standard deviation over three independent trials.

# hidden neurons	result type	SVI Train	SVI Valid	SVI Test	SGD Train	SGD Valid	SGD Test
128	Initial	39.64 (1.99)	39.52 (1.84)	39.83 (1.95)	6.95 (4.36)	7.05 (4.37)	6.91 (4.31)
	Final	63.55 (0.25)	63.44 (0.26)	63.47 (0.23)	51.62 (2.22)	51.38 (2.15)	51.63 (2.29)
256	Initial	52.02 (0.95)	51.84 (1.04)	52.02 (1.01)	23.38 (3.86)	23.35 (3.9)	23.43 (3.86)
	Final	66.56 (0.08)	66.2 (0.13)	66.26 (0.09)	59.24 (1.56)	59.14 (1.59)	59.1 (1.48)
512	Initial	57.88 (0.36)	57.57 (0.39)	57.68 (0.42)	33.55 (2.42)	33.46 (2.58)	33.64 (2.46)
	Final	69.12 (0.13)	68.52 (0.07)	68.72 (0.07)	64.28 (0.77)	63.99 (0.71)	64.07 (0.88)

References

[1] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pages 242–252. PMLR, 2019.

[2] Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332. PMLR, 2019.

[3] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems*, pages 8139–8148, 2019.

[4] Patrick L. Combettes, Jean-Christophe Pesquet, and Audrey Repetti. A variational inequality model for learning neural networks. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023. doi: 10.1109/ICASSP49357.2023.10095688.

[5] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

[6] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning*, pages 1675–1685. PMLR, 2019.

[7] Francisco Facchinei and Jong-Shi Pang. *Finite-dimensional variational inequalities and complementarity problems*. Springer, 2003.

[8] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[9] Bolin Gao and Lacra Pavel. On the properties of the softmax function with application in game theory and reinforcement learning. *arXiv preprint arXiv:1704.00805*, 2017.

[10] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

[11] Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. Ogb-lsc: A large-scale challenge for machine learning on graphs. *arXiv preprint arXiv:2103.09430*, 2021.

[12] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/ioffe15.html.

[13] Anatoli B. Juditsky and Arkadi S. Nemirovsky. Signal recovery by stochastic optimization. *Autom. Remote. Control.*, 80:1878–1893, 2019.

[14] David Kinderlehrer and Guido Stampacchia. *An introduction to variational inequalities and their applications*. SIAM, 2000.

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

[16] Georgios Kotsalis, Guanghui Lan, and Tianjiao Li. Simple and optimal methods for stochastic variational inequalities, i: Operator extrapolation. *SIAM Journal on Optimization*, 32(3): 2041–2073, 2022.

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

[18] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[19] Mingrui Liu, Hassan Rafique, Qihang Lin, and Tianbao Yang. First-order convergence theory for weakly-convex-weakly-concave min-max problems. *The Journal of Machine Learning Research*, 22(1):7651–7684, 2021.

[20] Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33): E7665–E7671, 2018.

[21] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6614.

[22] Jiri Outrata, Michal Kocvara, and Jochem Zowe. *Nonsmooth approach to optimization problems with equilibrium constraints: theory, applications and numerical results*, volume 28. Springer Science & Business Media, 2013.

[23] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS 2017 Workshop on Autodiff*, 2017. URL https://openreview.net/forum?id=BJJsrmfCZ.

[24] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy,

Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[25] Franco Pellegrini and Giulio Biroli. An analytic theory of shallow networks dynamics for hinge loss classification. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 5356–5367. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/3a01fc0853ebeba94fde4d1cc6fb842a-Paper.pdf.

[26] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1139–1147, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[27] Gilad Yehudai and Ohad Shamir. On the power and limitations of random features for understanding neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Appendix A. Proofs

To prove Proposition 3, we first bound the recovered parameters following techniques in [13].

Lemma 5 (Parameter recovery guarantee) *Under Assumption 2, suppose that there exists* $M < \infty$ *such that* $\forall \theta_L \in \boldsymbol{\theta}$,

$$\mathbb{E}_{X,Y[\theta_L]} \|\eta(\hat{X}_L) Y[\theta_L]\|_2 \leq M,$$

where $\mathbb{E}[Y[\theta_L]|X] = \phi(\eta(\hat{X}_L)\theta_L)$. *Choose adaptive step sizes* $\gamma = \gamma_t = [\kappa(t+1)]^{-1}$ *in* (5). *The sequence of estimates* $\{\hat{\theta}_L^{(T)}\}_{T\geq 1}$ *obeys the error bound*

$$\mathbb{E}_{\hat{\theta}_L^{(T)}} \{\|\hat{\theta}_L^{(T)} - \theta_L^*\|_2^2\} \leq \frac{4M^2}{\kappa^2(T+1)}. \tag{8}$$

Proof The proof employs classical techniques when analyzing the convergence of projection descent in stochastic optimization [13].

First, for any $\theta_L \in \boldsymbol{\theta}$,

$$\begin{aligned}
\mathbb{E}_{(X,Y[\theta_L])} \{\|\eta^\intercal(\hat{X}_L)\phi(\eta(\hat{X}_L)\theta_L)\|_2\} &= \mathbb{E}_X \{\|\mathbb{E}_{Y[\theta_L]}\{\eta(\hat{X}_L)Y[\theta_L]\}\|_2\} \\
&\leq \mathbb{E}_X \mathbb{E}_{Y[\theta_L]} \{\|\eta(\hat{X}_L)Y[\theta_L]\}\|_2\} \qquad \text{[Jensen's Inequality]} \\
&= \mathbb{E}_{(X,Y[\theta_L])} \{\|\eta(\hat{X}_L)Y[\theta_L]\}\|_2\} \leq M.
\end{aligned}$$

By the form of F, we then have that $\mathbb{E}_{X,Y}\{\|F(\theta_L)\|_2^2\} \leq 4M^2$ for any θ_L.

Next, note that each $\hat{\theta}_l^{(t)}$ is a deterministic function of $Z^N = \{(X_i, Y_i)\}_{i=1}^N$. Define the difference of estimation and its expected value as

$$D_t(Z^N) = \frac{1}{2}\|\hat{\theta}_l^{(t)} - \theta_L^*\|_2^2, \; d_t = \mathbb{E}_{Z^N}\{D_t(Z^N)\}.$$

As a result,

$$\begin{aligned} D_t(Z^N) &= \frac{1}{2}\| \operatorname{Proj}_{\boldsymbol{\theta}} \left[\hat{\theta}_l^{(t-1)} - \gamma_t F_{1:N}^T(\hat{\theta}_l^{(t-1)}) - \theta_L^*\right] \|_2^2 \\ &\leq \frac{1}{2}\|\hat{\theta}_l^{(t-1)} - \gamma_t F_{1:N}^T(\hat{\theta}_l^{(t-1)}) - \theta_L^*\|_2^2 \quad \text{[The projection is a contraction]} \\ &= \frac{1}{2}\|\hat{\theta}_l^{(t-1)} - \theta_L^*\|_2^2 - \gamma_t F_{1:N}^T(\hat{\theta}_l^{(t-1)})(\hat{\theta}_l^{(t-1)} - \theta_L^*) + \frac{1}{2}\gamma_t^2 \|F_{1:N}^T(\hat{\theta}_l^{(t-1)})\|_2^2. \end{aligned}$$

Taking expectation of both sides with respect to Z^N yields

$$\begin{aligned} d_t &\leq \frac{1}{2} d_{t-1} - \gamma_t \mathbb{E}_{Z^N}\left[F_{1:N}^T(\hat{\theta}_l^{(t-1)})(\hat{\theta}_l^{(t-1)} - \theta_L^*)\right] + 2\gamma_t^2 M^2 \\ &\leq (1 - 2\kappa\gamma_t) d_{t-1} + 2\gamma_t^2 M^2, \end{aligned}$$

where the last inequality follows by noting that $F_{1:N}$ is an unbiased estimator of F, which satisfies

$$F(\theta_L)^T(\theta_L - \theta_L^*) \geq \kappa \|\theta_L - \theta_L^*\|_2,$$

due to Assumption 2 on $F(\theta_L^*)$. Then, using triangle inequality yields the result.

Lastly, we prove by induction that if we define $R = (2M^2)/\kappa^2,\ \gamma_t = 1/\kappa(t+1)$, we have

$$d_t \leq \frac{R}{t+1}.$$

(Base case when $t = 0$.) Let B be the $\|\cdot\|_2$ diameter of $\boldsymbol{\theta}$ (e.g., $\|\theta_1 - \theta_2\|_2^2 \leq B^2\ \forall(\theta_1, \theta_2) \in \boldsymbol{\theta}$. Denote $\theta_L^+, \theta_L^- \in \mathcal{B}$ to satisfy $\|\theta_L^+ - \theta_L^-\|_2^2 = B^2$. By the definition of κ,

$$\langle F(\theta_L^+) - F(\theta_L^-), \theta_L^+ - \theta_L^- \rangle \geq \kappa \|\theta_L^+ - \theta_L^-\|_2^2 = \kappa B^2.$$

Meanwhile, the Cauchy-Schwarz inequality yields

$$\langle F(\theta_L^+) - F(\theta_L^-), \theta_L^+ - \theta_L^- \rangle = \langle \eta(\hat{X}_L)(\phi(\eta(\hat{X}_L))\theta_L^+) - \eta(\hat{X}_L)(\phi(\eta(\hat{X}_L))\theta_L^-), \theta_L^+ - \theta_L^- \rangle \leq 2MB.$$

Thus, $B \leq 2M/\kappa$. As a result, $B^2/2 \leq 2M^2/\kappa^2 = R$. Because $d_0 = \|\hat{\theta}_l^{(0)} - \theta_L^*\|_2^2 \leq B^2$,

$$d_0 \leq 2R = \frac{4M^2}{\kappa^2}.$$

(The inductive step from $t-1$ to t.) Note that by the definition of γ_t, $\kappa\gamma_t = 1/(t+1) \leq 1/2$. Thus,

$$\begin{aligned} d_t &\leq (1 - 2\kappa\gamma_t) d_{t-1} + 2\gamma_t^2 M^2 \\ &= \frac{R}{t}\left(1 - \frac{2}{t+1}\right) + \frac{R}{(t+1)^2} \leq \frac{R}{t+1}, \end{aligned}$$

whereby the proof is complete by the definition of d_t and R. ■

Proof [Proof of Proposition 3] Define

$$\tilde{\mathbb{E}}[Y_t | X_t] = \phi(\eta^\intercal(\hat{X}_{t,L})\theta_L^*), \tag{9}$$

which uses the true parameter θ_L^*. Note that when $p = 2$,

$$\begin{aligned}&\mathbb{E}_{X,\hat{\theta}_L^{(T)}}\{\|\hat{\mathbb{E}}[Y_t|X_t] - \mathbb{E}[Y_t|X_t]\|_2\}\\ \leq&\mathbb{E}_{X,\hat{\theta}_L^{(T)}}\{\underbrace{\|\hat{\mathbb{E}}[Y_t|X_t] - \tilde{\mathbb{E}}[Y_t|X_t]\|_2}_{(a)} + \underbrace{\|\tilde{\mathbb{E}}[Y_t|X_t] - \mathbb{E}[Y_t|X_t]\|_2}_{(b)}\}.\end{aligned}$$

We now bound (a) and (b) separately.
Bound of (a). We have that

$$\begin{aligned}E_{X,\hat{\theta}_L^{(T)}}\{\|\hat{\mathbb{E}}[Y_t|X_t] - \tilde{\mathbb{E}}[Y_t|X_t]\|_2\} &= \mathbb{E}_{X,\hat{\theta}_L^{(T)}}\{\|\phi(\eta^\intercal(\hat{X}_{t,L})\hat{\theta}_L^{(T)}) - \phi(\eta^\intercal(\hat{X}_{t,L})\theta_L^*)\|_2\}\\ &\leq \mathbb{E}_{X,\hat{\theta}_L^{(T)}}\{K\|\eta^\intercal(\hat{X}_{t,L})[\hat{\theta}_L^{(T)} - \theta_L^*]\|_2\}\\ &\leq K\lambda_{\max}(\eta(\hat{X}_{t,L})\eta^\intercal(\hat{X}_{t,L}))\mathbb{E}_{X,\hat{\theta}_L^{(T)}}\{\|\hat{\theta}_L^{(T)} - \theta_L^*\|_2\}.\end{aligned}$$

We can then use the bound on $\mathbb{E}_{\hat{\theta}^{(T)}}\{\|\hat{\theta}^{(T)} - \theta\|_2\}$ from the previous lemma to complete the proof. In addition, because p-norm is decreasing in p, we have that the bound holds for any $p \in [2, \infty]$. ■

Bound of (b). We have by Assumptions 1 and 2 that

$$\begin{aligned}E_{X,\hat{\theta}_L^{(T)}}\{\|\tilde{\mathbb{E}}[Y_t|X_t] - \mathbb{E}[Y_t|X_t]\|_2\} &= E_{X,\hat{\theta}_L^{(T)}}\{\phi(\eta^\intercal(\hat{X}_{t,L})\theta_L^*) - \phi(\eta^\intercal(X_{t,L}^*)\theta_L^*)\}\\ &\leq E_{X,\hat{\theta}_L^{(T)}}\{K\|\eta^\intercal(\hat{X}_{t,L})\theta_L^* - \eta^\intercal(X_{t,L}^*)\theta_L^*\|_2\}\\ &\leq E_X\{KDB\|\hat{X}_{t,L} - X_{t,L}^*\|_2\}\\ &\leq KDB\epsilon.\end{aligned}$$

Proof [Proof of Proposition 4] The crux is to bound the expected value of the norm of F evaluated at the stochastic OE estimate. This bound results from [16, Proposition 3.8], where in general, for any $\theta_L \in \boldsymbol{\theta}$, we can use the residual

$$\mathbb{E}[\text{res}(\theta_L)] \leq \delta,\ \text{res}(\theta_L) = \min_{y \in -N_{\boldsymbol{\theta}}(\theta_L)} \|y - F(\theta_L)\|_2$$

as the termination criteria for the recurrence under a certain choice of the Bregman's distance $V(a, b)$; we let $V(a, b) = \|a - b\|_2^2/2$ in our case. The quantity $N_{\boldsymbol{\theta}}(\theta_L) = \{y \in \mathbb{R}^p | \langle y, \theta' - \theta_L\rangle, \forall \theta' \in \boldsymbol{\theta}\}$ denotes the normal cone of $\boldsymbol{\theta}$ at θ_L. Then, under the assumptions on F and choices of step sizes, we can restate [16, Proposition 3.8] in our special case as

$$\mathbb{E}_{\hat{\theta}_l^{(R)}}[\text{res}(\hat{\theta}_l^{(R)})] \leq \frac{3\sigma}{\sqrt{T}} + \frac{12K_2\sqrt{2\|\theta_L^*\|_2^2 + \frac{2\sigma^2}{L^2}}}{\sqrt{T}}.$$

When we assume $\boldsymbol{\theta}$ is the entire space, $N_{\boldsymbol{\theta}}(\theta_L) = \{\mathbf{0}\}$ whereby $\text{res}(\hat{\theta}_l^{(R)}) = \|F(\hat{\theta}_l^{(R)})\|_2$.

Furthermore, note that for any matrix $A \in \mathbb{R}^{m\times n}$ and vectors $x, x' \in R^n$, we have

$$\|x - x'\|_2 \leq \|A(x - x')\|_2/\sigma_{\min}(A),$$

Table 2: Hyper-parameter selection for each experiment. The exact same hyper-parameters are used for `SVI` vs. SGD.

Dataset	Train & Test size	Batch size	Epochs	Learning rate
One-layer probit model	2000 train, 500 test	200	200	0.005
Two-layer GCN model	2000 train, 2000 test	100	200	0.001
Solar ramping event	365 train, 365 test	30	100	0.001
Traffic anomaly detection	6138 train, 2617 test	600	100	0.001
OGB node classification	90941 training nodes, 29799 validation nodes, 48603 test nodes	all training nodes	1000	0.001
MNIST classification	60000 train, 10000 test	128	20	0.005
CIFAR-10 classification	50000 train, 10000 test	128	200	0.005

where $\sigma_{\min}(A)$ denotes the smallest singular value of A. As a result, by letting $A = \eta(\hat{X}_L), x = \hat{\mathbb{E}}[Y_t|X_t], x' = \mathbb{E}[Y_t|X_t]$ we have in expectation that

$$\mathbb{E}_{\hat{\theta}_l^{(R)}}\{\|\mathbb{E}_X\{\sigma_{\min}(\eta(\hat{X}_L))[\hat{\mathbb{E}}[Y_t|X_t] - \mathbb{E}[Y_t|X_t]\}\|_p\} \leq \mathbb{E}_{\hat{\theta}_l^{(R)}}\|F(\hat{\theta}_l^{(R)})\|_2 = \mathbb{E}_{\hat{\theta}_l^{(R)}}[\text{res}(\hat{\theta}_l^{(R)})],$$

where we used the fact $F(\hat{\theta}_l^{(R)}) = \mathbb{E}_{X,Y}\{\eta^\intercal(\hat{X}_L)[\phi_L(\eta(\hat{X}_L)\hat{\theta}_l^{(R)}) - Y]\} = \mathbb{E}_X\{\eta^\intercal(\hat{X}_L)[\hat{\mathbb{E}}[Y_t|X_t] - \mathbb{E}[Y_t|X_t]]]\}$. ■

Appendix B. Additional experiments

B.1. Hyper-parameter choices

All implementation are done using `PyTorch` [24] and `PyTorch Geometric` [8] (for GNN). To ensure fair comparisons, we ensure the following hyperparameters are *identical* to both `SVI`-based and gradient-based methods in each experiment; Table 2 provides the complete list of hyper-parameter selection for each experiment.

- Data: (a) the size of training and test data (b) batch (batch size and samples in mini-batches).
- Model: (a) architecture (e.g., layer choice, activation function, hidden neurons) (b) loss function in empirical risk minimization.
- Training regime: (a) parameter initialization (b) hyperparameters for optimizers (e.g., learning rate) (c) total number of epochs.

In short, all except the way gradients are defined are kept the same for each comparison—our proposed `SVI` implements the "skipping" idea in Algorithm 1 and SGD uses the gradient of the loss with respect to parameters in each hidden layer.

Table 3: Two-layer GCN model on random graphs with increasing number of graph nodes. We show the training and test ℓ_2 error as defined in (10) along the training epochs. Entries in brackets indicates standard deviation over 3 independent initialization of model parameters.

	Epoch 50				Epoch 100				Epoch 200			
# nodes	SGD train	SVI train	SGD test	SVI test	SGD train	SVI train	SGD test	SVI test	SGD train	SVI train	SGD test	SVI test
15	0.110	0.104	0.107	0.101	0.099	0.089	0.097	0.087	0.084	0.073	0.082	0.071
	(2.01e-2)	(1.72e-2)	(1.63e-2)	(1.34e-2)	(1.79e-2)	(1.21e-2)	(1.46e-2)	(9.09e-3)	(1.43e-2)	(5.81e-3)	(1.17e-2)	(4.11e-3)
40	0.102	0.096	0.101	0.095	0.092	0.083	0.092	0.083	0.079	0.069	0.079	0.068
	(1.60e-2)	(1.39e-2)	(1.60e-2)	(1.38e-2)	(1.45e-2)	(1.04e-2)	(1.47e-2)	(1.05e-2)	(1.21e-2)	(5.76e-3)	(1.26e-2)	(6.20e-3)
100	0.092	0.087	0.093	0.088	0.085	0.078	0.086	0.079	0.075	0.066	0.077	0.067
	(1.27e-2)	(1.13e-2)	(1.37e-2)	(1.22e-2)	(1.18e-2)	(9.29e-3)	(1.30e-2)	(1.03e-2)	(1.06e-2)	(6.55e-3)	(1.19e-2)	(7.75e-3)
300	0.080	0.077	0.081	0.077	0.076	0.070	0.077	0.071	0.069	0.060	0.070	0.061
	(9.48e-3)	(8.80e-3)	(1.04e-2)	(9.68e-3)	(9.09e-3)	(7.86e-3)	(1.01e-2)	(8.78e-3)	(8.61e-3)	(6.50e-3)	(9.67e-3)	(7.49e-3)
600	0.073	0.070	0.074	0.071	0.070	0.064	0.070	0.065	0.064	0.056	0.065	0.056
	(7.63e-3)	(7.34e-3)	(8.32e-3)	(7.98e-3)	(7.43e-3)	(6.86e-3)	(8.14e-3)	(7.54e-3)	(7.23e-3)	(6.05e-3)	(7.97e-3)	(6.80e-3)

B.2. Training two-layer GCN

We compare SVI and SGD when training a two-layer GCN model on graphs of varying sizes. We vary the number of nodes $n \in \{15, 40, 100, 300, 600\}$. Each graph $G = (\mathcal{V}, \mathcal{E})$ has an edge probability of 0.15 between any two nodes i, j in $\mathcal{V}$. For the graph data, we generate node labels $Y_i \in \{0, 1\}^n$ based on an input signal $X_i \in \mathbb{R}^{n\times 2}$, whose entries are i.i.d. samples from $\mathcal{N}(0, 1)$. Specifically, we design the neural network in (3) that represents $\mathbb{E}[Y_i|X_i, \theta]$ as a two-layer GCN with two hidden nodes, ReLU activation, and sigmoid output function, Entries of the true parameter θ^* are i.i.d. samples from $\mathcal{N}(1, 1)$.

After training the model with SVI or SGD to obtain $\hat{\theta}$, we compare the error in ℓ_2 model recovery of $\mathbb{E}[Y_i|X_i, \theta^*]$ on M test samples (X_i, Y_i):

$$\ell_2 \text{ model recovery error} = M^{-1} \sum_{i=1}^{M} \|\mathbb{E}[Y_i|X_i, \hat{\theta}] - \mathbb{E}[Y_i|X_i, \theta^*]\|_2. \tag{10}$$

Table 3 shows that SVI consistently yields smaller training and test ℓ_2 errors along the training trajectory across different graph sizes.

To better understand how SVI update parameters, Figure 3 zooms in the dynamics for 16 neurons (right figures in (a) and (b)) and shows the corresponding ℓ_2 model recovery errors (left figures in (a) and (b)). To show neuron dynamics, we follow the visualization techniques in [25, Figure 2]. Specifically, we plot the norm of first-layer neuron weights, where the norm is defined in terms of the

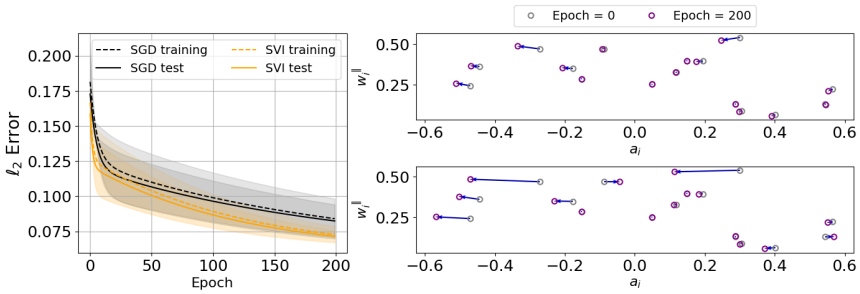


Figure 3: Two-layer GCN model recovery error and neuron dynamics visualization. Left: ℓ_2 error in (10). Right: visualization of the training dynamics by SGD (top) and SVI (bottom).

Table 4: Solar ramping event detection under varying sizes of the GCN. Entries in brackets indicates standard deviation over 3 independent initialization of model parameters.

	MSE Loss				Classification error				Weighted F_1 score			
# hidden neurons	SGD Training	SVI Training	SGD Test	SVI Test	SGD Training	SVI Training	SGD Test	SVI Test	SGD Training	SVI Training	SGD Test	SVI Test
32	0.224 (4.67e-3)	0.200 (9.95e-4)	0.223 (2.40e-3)	0.204 (1.30e-3)	0.297 (5.88e-3)	0.275 (4.54e-3)	0.333 (1.28e-2)	0.296 (1.13e-2)	0.704 (6.62e-3)	0.727 (4.36e-3)	0.659 (1.46e-2)	0.704 (1.15e-2)
64	0.213 (7.66e-4)	0.191 (8.49e-4)	0.211 (1.67e-3)	0.195 (1.12e-3)	0.283 (4.33e-3)	0.263 (1.59e-3)	0.308 (1.04e-2)	0.272 (1.15e-3)	0.719 (4.57e-3)	0.737 (1.59e-3)	0.689 (1.16e-2)	0.728 (1.14e-3)
128	0.219 (1.79e-3)	0.195 (6.22e-4)	0.217 (1.13e-3)	0.199 (5.14e-4)	0.295 (3.15e-3)	0.267 (2.43e-3)	0.328 (6.13e-3)	0.282 (9.04e-4)	0.706 (3.50e-3)	0.734 (2.41e-3)	0.664 (7.73e-3)	0.718 (9.11e-4)

inner product with initial weights, against that of second-layer neuron weights. One circle represents one neuron, with arrows representing the direction of movement along the initial weights. We then connect the initial and final dots to indicate the displacement of neurons. In terms of results, after 200 epochs, SVI displaces the neurons from their initial position further, where such displacement leads to faster convergence in practice.

B.3. Solar ramping event detection

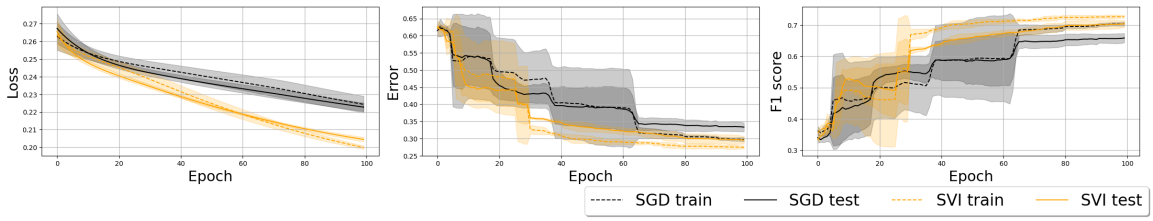
In this experiment, the goal is to identify ramping events within a network of solar sensors, where ramping events are defined over abrupt changes of the sensor inputs. The raw solar radiation data are retrieved from the National Solar Radiation Database for 2017 and 2018. We consider data from 10 city downtown in California, where each city or location is a node in the network. The goal is to identify ramping events daily, which are abrupt changes in the solar power generation. Thus, $Y_{t,i} = 1$ if node i at day t experiences a ramping event. We define feature $X_t = \{Y_{t-1} \dots, Y_{t-w}\}$ as the collection of past w days of observation and pick $w = 5$. We estimate edges via a k-nearest neighbor approach based on the correlation between training ramping labels, with $k = 4$. Data in 2017 are used for training ($N = 360$) and the rest for testing ($N_1 = 365$), and we let $B = 30$ and $E = 100$.

We train three-layer GCN models of varying sizes (i.e., number of hidden nodes). Table 4 shows that SVI consistently reaches lower test classification error and higher test weighted F_1 scores[2]. Figure 4 shows faster intermediate convergence results by SVI in terms of both metrics.

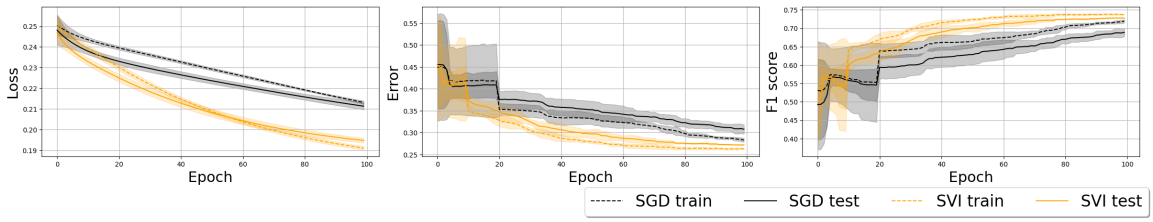
B.4. Traffic flow anomaly detection

In this experiment, the goal is to identify multi-class bi-hourly anomalous traffic flow observations in a sensor network. The raw bi-hourly traffic flow data are from the California Department of Transportation, where we collected data from 20 non-uniformly spaced traffic sensors in 2020 from https://pems.dot.ca.gov/. Data are available hourly, with $Y_{t,i} = 0$ denoting normal observations. Meanwhile, $Y_{t,i} = 1$ (resp. 2) if the current traffic flow lies outside the upper (resp. lower) 90% quantile over the past four days of traffic flow of its nearest four neighbors based on sensor proximity. Feature X_t contains past w days of observation and set $w = 4$, where the edges include the nearest five neighbors based on sensor locations. Data in the first nine months are training data and the rest for testing.

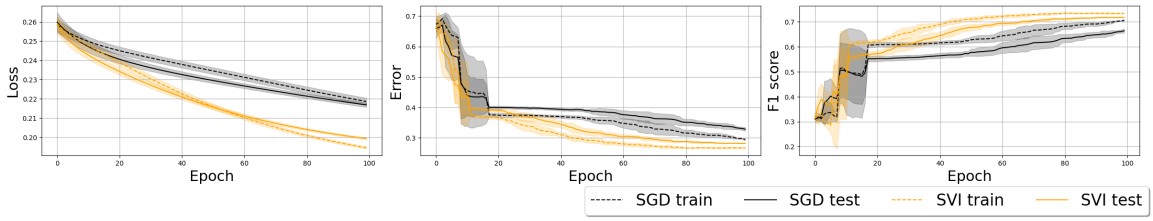
[2]The F_1 scores are weighted by support (the number of true instances for each label).



(a) 32 hidden neurons. Left to right: MSE loss, classification error, and weighted F_1 score.



(b) 64 hidden neurons. Left to right: MSE loss, classification error, and weighted F_1 score.



(c) 128 hidden neurons. Left to right: MSE loss, classification error, and weighted F_1 score.

Figure 4: Solar ramping event detection under various hidden neurons. Results are plotted with one standard error bars over three independent trials.

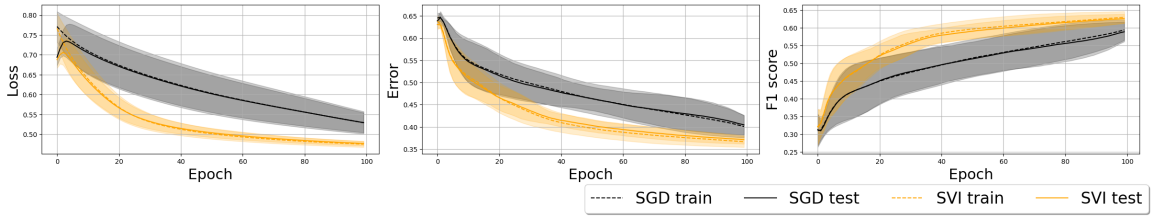
We train three-layer GCN models of varying sizes. Table 5 shows that `SVI` consistently reaches lower test classification error and higher test weighted F_1 scores. Figure 5 shows faster intermediate convergence results by `SVI` in terms of both metrics.

B.5. Image classification

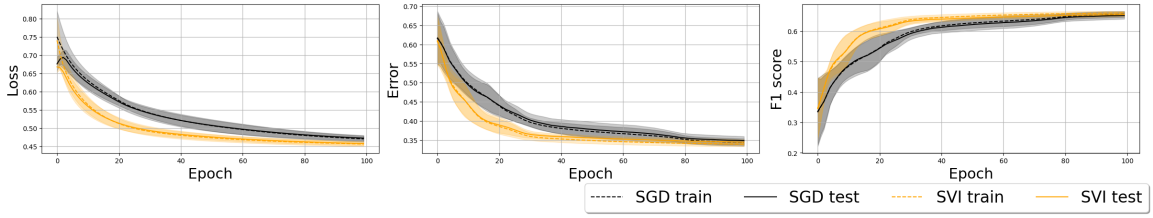
We compare both methods on training image classifiers. Specifically, we train the LeNet [18] on MNIST [5] and the VGG-16 on CIFAR-10 [17]. On MNIST, we train the initial 10% of total

Table 5: Traffic data multi-class anomaly detection under varying sizes of the GCN. Entries in brackets indicates standard deviation over 3 independent initialization of model parameters.

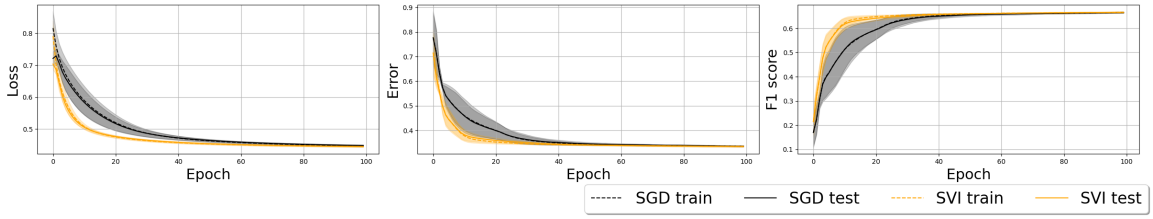
	MSE Loss				Classification error				Weighted F_1 score			
# hidden neurons	SGD Training	`SVI` Training	SGD Test	`SVI` Test	SGD Training	`SVI` Training	SGD Test	`SVI` Test	SGD Training	`SVI` Training	SGD Test	`SVI` Test
32	0.529 (2.84e-2)	0.475 (7.64e-3)	0.529 (2.51e-2)	0.477 (5.63e-3)	0.401 (2.49e-2)	0.367 (1.24e-2)	0.404 (2.22e-2)	0.371 (9.59e-3)	0.594 (2.89e-2)	0.629 (1.67e-2)	0.589 (2.76e-2)	0.626 (1.29e-2)
64	0.471 (8.25e-3)	0.457 (6.51e-3)	0.473 (7.67e-3)	0.458 (5.19e-3)	0.344 (9.52e-3)	0.339 (5.86e-3)	0.349 (1.12e-2)	0.345 (6.66e-3)	0.655 (9.91e-3)	0.660 (6.05e-3)	0.651 (1.14e-2)	0.655 (6.71e-3)
128	0.447 (2.70e-3)	0.445 (1.84e-3)	0.448 (2.44e-3)	0.445 (1.98e-3)	0.334 (1.64e-3)	0.334 (2.00e-3)	0.335 (2.03e-3)	0.334 (3.27e-3)	0.665 (1.77e-3)	0.665 (1.97e-3)	0.664 (2.16e-3)	0.666 (3.36e-3)



(a) 32 hidden neurons. Left to right: MSE loss, classification error, and weighted F_1 score.



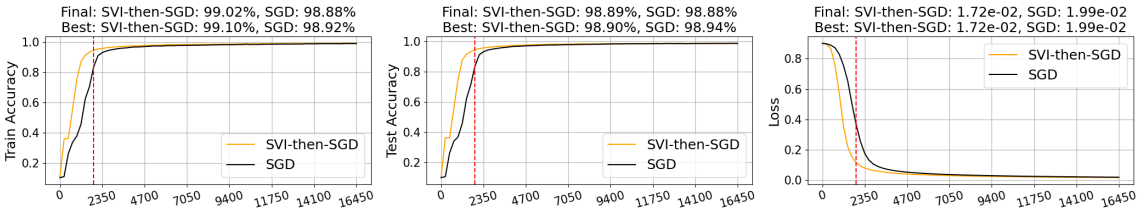
(b) 64 hidden neurons. Left to right: MSE loss, classification error, and weighted F_1 score.



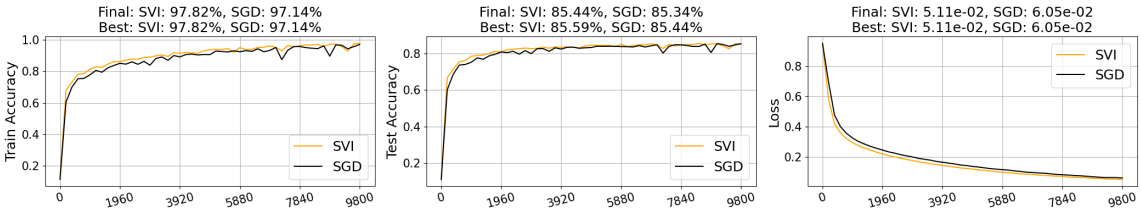
(c) 128 hidden neurons. Left to right: MSE loss, classification error, and weighted F_1 score.

Figure 5: Traffic data multi-class anomaly detection under various hidden neurons. Results are plotted with one standard error bars over three independent trials.

training batches by `SVI` and the rest 90% by SGD, and we call this hybrid technique `SVI`-then-SGD. Figure 6 shows training and test metrics over training batches. On CIFAR-10, `SVI` consistently yields higher accuracy and lower less than SGD. On MNIST, the hybrid approach shows faster initial convergence than SGD due to the use of `SVI` and also reaches higher accuracy and lower loss by the end of all training batches.



(a) MNIST by LeNet: training accuracy (left), test accuracy (middle), and training loss (right).



(b) CIFAR10 by VGG-16: training accuracy (left), test accuracy (middle), and training loss (right).

Figure 6: Classification accuracies and training losses by both methods. We plot the metrics over training batches in each sub-figure. In the title, "Final" represents the metric at the end of all training epochs and "Best" represents the highest/lowest metric throughout training epochs. On MNIST, training before the dashed dotted red line is by `SVI`, and we continue train the SVI-warm-started model afterwards by SGD.