An alternative approach to train neural networks using monotone variational inequality

Anonymous authors
Paper under double-blind review

Abstract

The current paper investigates 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 a recent work of (Juditsky & Nemirovsky, 2019). 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 fully-connected neural networks, graph neural networks (GNN), and convolutional networks (CNN) (`SVI` is completely general for training other network architectures). We show the competitive or better performance of `SVI` compared to widely-used stochastic gradient descent methods on both synthetic and real network data prediction tasks regarding various performance metrics, especially in the improved efficiency in the early stage of training.

1 Introduction

Neural network (NN) training (Sutskever et al., 2013; Kingma & Ba, 2015; Ioffe & Szegedy, 2015; Yehudai & Shamir, 2019) 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 (Neyshabur et al., 2014; Mei et al., 2018; Arora et al., 2019a;b; Allen-Zhu et al., 2019; Du et al., 2019). 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 (Juditsky & Nemirovsky, 2019) presented a somehow surprising result that some non-convex issues can be circumvented in special cases by problem reformulation. In particular, it was shown that one can formulate the parameter estimation problem of the generalized linear models (GLM) as solving an 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 nor model recovery. Thus, the formulation through MVI leads to strong performance guarantees and computationally efficient procedures.

In this paper, inspired by (Juditsky & Nemirovsky, 2019) and the fact that certain GLM (such as logistic regression) can be viewed as the simplest network with only one layer, we consider a new scheme for neural network training based on MVI. This is a drastic departure from the widely used gradient descent algorithm for neural network training — we replace the gradient of a loss function with operators inspired by MVI theory. The benefits include (i) in special cases, we can establish strong training and prediction guarantees in Section 4; (ii) for general cases, through extensive numerical studies on synthetic and real-data in Section 5, we demonstrate the faster convergence to a local solution by our approach relative to gradient descent in a comparable setup.

To the best of our knowledge, the current paper is the first to study MVI for training neural networks. Our `SVI`, as a general way of modifying the parameter update scheme in NN training, can be readily applied to various deep architectures. Our `SVI` is a general way of modifying the parameter update scheme in NN training and can be readily applied to various deep architectures. In this work, beyond fully-connected (FC) neural networks, we especially study node classification in graph neural networks (GNN) (Wu et al., 2019; Pilanci & Ergen, 2020) and image classification. Our technical contributions include:

- Develop a general and practical algorithm called *stochastic variational inequality* (`SVI`). In particular, the algorithm provides a fundamentally different but easy-to-implement alternative to using gradient of the original loss objective with respect to parameters. In addition, `SVI` and gradient-based methods barely differ in computation.
- Reformulate the single-layer neural network training as solving a monotone variational inequality, with guarantees on recovery and prediction accuracy. For multi-layer networks, similar guarantees are obtained assuming ϵ-approximate output from all except the last layer.
- Compare `SVI` with widely-used stochastic gradient descent methods to demonstrate that `SVI` is flexible on various tasks and competitive against SGD or Adam, especially the improved efficiency in the early stage of training.

Literature. MVI has been studied mainly in the optimization problem context (Kinderlehrer & Stampacchia, 1980; Facchinei & Pang, 2003). More recently, VI has been used to solve min-max problems in Generative Adversarial Networks (Lin et al., 2018) and reinforcement learning (Kotsalis et al., 2020). Our theory and techniques are inspired by (Juditsky & Nemirovsky, 2019), but we offer a thorough investigation of using MVI to train multi-layer NN. In retrospect, our techniques bear similarity to works on “matching loss“ (Amid et al., 2022) but are fundamentally different as we leverage MVI theory and lead to performance guarantees. In addition, we differ from (Pilanci & Ergen, 2020), which views two-layer NNs as convex regularizers: we do not convexify the loss minimization, and our `SVI` extends beyond two-layer networks.

2 Problem Setup

We briefly describe notation of general NN and GNN of interest and provide preliminaries of MVI.

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

$$\mathbb{E}[Y|X, \Theta] := G(X, \Theta) = \phi_L(g_L(X_L, \Theta_L)), \tag{1}$$

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, $\Theta = \{\Theta_1, \dots, \Theta_L\}$ denotes model parameters, and each ϕ_l denotes the activation function at layer l. In particular, assume there exists Θ^* so that $\mathbb{E}[Y|X] = f(X, \Theta^*)$. Our GNN notations are standard and are described in Appendix B.1.

In practice, our MVI framework to be introduced next applies to *any* form of $g_l(X_l, \Theta_l)$ in (1) for $l = 1, \dots, L$. If $g_L(X_L, \Theta_L)$ at the last output layer has a particular linear form, we can also obtained theoretical guarantees under idealized assumptions.

2.2 MVI Preliminaries. Given a parameter set $\mathbf{\Theta} \subset \mathbb{R}^p$, we call a continuous mapping (operator) $F : \mathbf{\Theta} \to \mathbb{R}^p$ *monotone* if for all $\Theta_1, \Theta_2 \in \mathbf{\Theta}$, $\langle F(\Theta_1) - F(\Theta_2), \Theta_1 - \Theta_2 \rangle \geq 0$ (Juditsky & Nemirovsky, 2019). The operator is called *strongly monotone* with modulus κ if for all $\Theta_1, \Theta_2 \in \mathbf{\Theta}$,

$$\langle F(\Theta_1) - F(\Theta_2), \Theta_1 - \Theta_2 \rangle \geq \kappa \|\Theta_1 - \Theta_2\|_2^2. \tag{2}$$

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

$$\langle F(\bar{\Theta}), \Theta - \bar{\Theta} \rangle \geq 0. \qquad \text{VI}[F, \mathbf{\Theta}] \tag{3}$$

It is known that if $\mathbf{\Theta}$ is compact, then (3) has at least one solution (Outrata et al., 2013, Theorem 4.1). In addition, if $\kappa > 0$ in (2), then (3) has exactly one solution (Outrata et al., 2013, Theorem 4.4). Under mild computability assumptions, the solution can be efficiently solved to high accuracy, using various iterative schemes (Juditsky & Nemirovsky, 2019).

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

3 MVI FOR NEURAL NETWORK TRAINING

3.1 Single-layer training. Let us first consider a single-layer neural network (i.e., $L = 1$ in (1)) with a particular form of pre-activation: $g(X, \Theta) = \eta(X)\Theta$ for an arbitrary feature transformation η. Although this form can be overly simplistic, it is typically the last layer of a deep neural network. For FC networks, $\eta(X) = X$, i.e., the identity map. For GNN models, $\eta(X) = \sum_{r=1}^{R} h_r(L_g)X$ as the sum of R fixed graph filters determined by graph Laplacian L_g (Kipf & Welling, 2017; Defferrard et al., 2016; Hamilton et al., 2017). One can also write down the corresponding formulation for convolutional layers in image classification under more complex notation. Note that this is mathematically equivalent to a GLM (Juditsky & Nemirovsky, 2019).

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 $\eta(X)$. We will explain a few properties of F in Sec. 4, Lemma 4.1. Denote $\widehat{F}$ as the empirical sample version of F. Then the typical training based on MVI is the projected gradient descent

$$\Theta \leftarrow \text{Proj}_{\mathbf{\Theta}}(\Theta - \gamma\widehat{F}(\Theta)). \tag{5}$$

where $\gamma > 0$ is the step-size and $\text{Proj}_{\mathbf{\Theta}}$ projects back estimates to the feasible parameter domain $\mathbf{\Theta}$.

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 $\widehat{F}$. In particular, $\widehat{F}$ needs not correspond to the gradient of any loss function. Nevertheless, we will show in Section 4.3, Proposition 4.8 that $\widehat{F}$ is exactly the gradient of the cross-entropy loss, whereby (5) coincides with SGD.

3.2 Multi-layer training. Our goal is to train multi-layer networks based on (4). Specifically, we want to obtain $F_l(\Theta_l)$ for parameters Θ_l in layer l of the neural network in (1). These $\{F_l(\Theta_l)\}_{l=1}^{L}$ are then used during optimization (e.g., via projected gradient descent (5) or Adam (Kingma & Ba, 2015)).

However, there are several fundamental difficulties of extending $F(\Theta)$ in (4) to training generic multi-layer neural networks. First, (4) requires that the pre-activation mapping $\eta(X)\Theta$ is linear in Θ. This linearity assumption does not hold for many network layer types. Second, more importantly, (4) is defined using Y, the observation of response variable. However, such observation is unavailable when we define $F_l(\Theta_l)$ for intermediate layers, where Y_l denotes the response after the l-th hidden layer. Third, note that (4) does not depend on any loss objective (e.g., mean-squared error, cross-entropy loss, etc.), so that the extension under a specific loss objective when training the network is unclear.

Despite the difficulties, we propose an MVI-based scheme to train generic neural networks under any loss objective. To motivate the scheme, we first present a mathematically equivalent view of $F(\Theta)$ in (4). Assume we have the mean-squared-error (MSE) loss $\mathcal{L}(\hat{Y}, Y) := 1/2\|\hat{Y} - Y\|_2^2$ and the linear pre-activation mapping $\widetilde{X} := \eta(X)\Theta$. Denote $\hat{Y} := \phi(\widetilde{X})$ as the prediction. By chain rule

$$\begin{aligned} F(\Theta) :&= \mathbb{E}_{X,Y}\{\eta^{\intercal}(X)[\phi(\eta(X)\Theta) - Y]\} \\ &= \mathbb{E}_{X,Y}\{\nabla_{\hat{Y}}\mathcal{L} \circ \nabla_{\Theta}\widetilde{X}\}. \end{aligned} \tag{6}$$

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 output. The second term $\nabla_{\Theta}\widetilde{X}$ is the gradient of the *pre-activation* with respect to the parameter. In particular, (6) is well-defined for any loss objective $\mathcal{L}$ and at any hidden layer l—we would simply replace $\widetilde{X}$ with $\widetilde{X}_l$, the pre-activation mapping at the l-th layer.

Therefore, based on the observation (6), `SVI` in Algorithm 1 heuristically extends MVI to training any neural network described by (1); Figure 1 illustrates the idea. Notation-wise, the subscript $_i$ denotes sample index and the subscript $_l$ denotes the output of first l network layers. In retrospect, comparing with the commonly used gradient $\nabla_{\Theta}\mathcal{L}$, Algorithm 1 only differs by *skipping* the additional computation $\nabla_{\eta(X)\Theta}\hat{Y} = \nabla_{\eta(X)\Theta}\phi(\eta(X)\Theta)$. This operation is precisely the derivative of the point-wise non-linearity ϕ. Therefore, `SVI` barely differs in terms of computational cost against gradient-based methods. In practice, this skipping leads to a different neuron update dynamics—see Remark 3.1 for details.

There are several important benefits of Algorithm 1. First, it addresses the aforementioned challenges of extending MVI to multi-layer training—the Algorithm is applicable to arbitrary form of $g_l(X_l, \Theta_l)$ and loss function $\mathcal{L}$. It requires no observation of responses from hidden layers. Second, it is easy to implement upon leveraging the benefit of automatic differentiation (Paszke et al., 2017). We implement the skipping idea via backpropagating another loss $\widetilde{\mathcal{L}}(\Theta_l)$ as in line 4 on the parameters. This loss $\widetilde{\mathcal{L}}(\Theta_l)$ is simple to compute, as the quantity $\widetilde{X}_{l+1}$ is available during the forward pass on data and the gradient $\nabla_{X_{l+1}}\mathcal{L}$ is available upon backpropagating the original loss with respect to output of the current layer l. Third, SVI is very flexible as one can apply SVI to all or a subset of layers in the network.

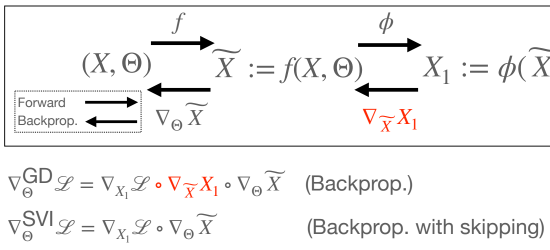

f
ϕ
(X,Θ)
$\widetilde{X} := f(X,\Theta)$
$X_1 := \phi(\widetilde{X})$
$\cdots$
$\mathscr{L}$
Forward
Backprop.
$\nabla_\Theta \widetilde{X}$
$\nabla_{\widetilde{X}} X_1$
$\nabla_{X_1}\mathscr{L}$
$\nabla^{\text{GD}}_\Theta \mathscr{L} = \nabla_{X_1}\mathscr{L} \circ \nabla_{\widetilde{X}} X_1 \circ \nabla_\Theta \widetilde{X}$ (Backprop.)
$\nabla^{\text{SVI}}_\Theta \mathscr{L} = \nabla_{X_1}\mathscr{L} \circ \nabla_\Theta \widetilde{X}$ (Backprop. with skipping)

Figure 1: Gradient descent (GD) vs. SVI: the difference appears in the skipping of differentiation with respect to the point-wise non-linearity ϕ. Details are in Algorithm 1.

Remark 3.1 (Effect on training dynamics). We remark a key difference between parameter update in SGD and in SVI, which ultimately affects training dynamics. Suppose the activation function is ReLU. It is well-known that SGD does not update weights of *inactive neurons* because the gradient of ReLU with respect to them is zero. However, SVI does not discriminate as it *skips* this derivative computation. Thus, one can expect that SVI results in further weight updates than SGD, which experimentally seems to speed up the initial model convergence. We illustrate this phenomenon in Figure 4 and will provide explanations in future works.

Algorithm 1 Stochastic variational inequality (SVI).

```
Require: Inputs are (a) Training data $\{(X_i,Y_i)\}_{i=1}^N$ (b) An $L$-layer network $G(X,\Theta) := \{\phi_l(g_l(X_l,\Theta_l))\}_{l=1}^L$ (c) Boolean mask $BM$ of length $L$ (d) Loss function $\mathcal{L} := \mathcal{L}(G(X,\Theta),Y)$
Ensure: Estimated parameters $\hat{\Theta} := \{\hat{\Theta}_l\}_{l=1}^L$
 1: while Training not stopped do
 2:   for Layer $l = L,\dots,1$ do
 3:     if $BM[l]$ is True then {▷ Use SVI}
 4:       Compute $F_l(\Theta_l) := \nabla_{\Theta_l}\widetilde{\mathcal{L}}(\Theta_l)$ over mini-batches, where
          (a). $\widetilde{\mathcal{L}}(\Theta_l) := \sum_{x\in \boldsymbol{X_l}} \boldsymbol{X_l}$ {▷ Sum over all elements of the tensor $\boldsymbol{X_l}$}
          (b). $\boldsymbol{X_l} := \widetilde{X}_{l+1} \odot \nabla_{X_{l+1}}\mathcal{L}$ {▷ Use element-wise product $\odot$}
          (c). $\widetilde{X}_{l+1} := g_l(X_l,\Theta_l)$ and $X_{l+1} := \phi_l(\widetilde{X}_{l+1})$ as in (1)
 5:       $\widehat{\Theta}_l = \widehat{\Theta}_l - \eta\nabla_{\Theta_l}F_l(\Theta_l)$.[2]
 6:     else {▷ Use gradient descent}
 7:       $\widehat{\Theta}_l = \widehat{\Theta}_l - \eta\nabla_{\Theta_l}\mathcal{L}$.
 8:     end if
 9:   end for
10: end while
```

4 Guarantee of model recovery by MVI

We now present training and estimation guarantees on model recovery for the *last-layer training* when $g_L(X_L,\Theta_L) = \eta_L(X_L)\Theta_L$ in (1) and previous layers are estimated with ϵ accuracy.

Let $\{(X_i,Y_i)\}_{i=1}^N$ be N training data from model (1) with $\Theta = \Theta^*$. Let $f^*_{1:L-1}(\cdot)$ and $\hat{f}_{1:L-1}(\cdot)$ be the oracle and estimated mapping from the previous $L-1$ layers, and let $X^*_{i,L}$ and $\hat{X}_{i,L}$ be the corresponding outputs from the mappings. For the subsequent theoretical analyses, we will assume that the expected ℓ_p difference of these two mappings are close enough (see Assumption 4.2). In this setting, $\eta(X)$ thus becomes $\eta(\hat{X}_L)$ for a generic feature X. For the operator $F(\Theta_L)$ in (4), where $\boldsymbol{\Theta}$ is the parameter space of Θ_L, consider its empirical average $F_{1:N}(\Theta_L) := (1/N)\sum_{i=1}^N F_i(\Theta_L)$, where $F_i(\Theta_L) := \eta^\intercal(\hat{X}_{i,L})[\phi(\eta(\hat{X}_{i,L})\Theta_L) - Y]$. Let $\widehat{\Theta}_L^{(T)}$ be the estimated parameter after T

[2] One can use any alternative optimizer, such as Adam (Kingma & Ba, 2015).

training iterations, using $F_{1:N}$ and (5). For a test feature Y_t and the transformation $\hat{X}_{t,L}$, consider

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

which is the estimated prediction using $\widehat{\Theta}_L^{(T)}$ and will be measured against the true model $\mathbb{E}[Y_t|Y_t]$ under Θ_L^* and $X_{t,L}^*$. In particular, we will provide error bound on $\|\widehat{\mathbb{E}}[Y_t|Y_t] - \mathbb{E}[Y_t|Y_t]\|_p, p \geq 2$ which crucially depends on the strong monotonicity modulus κ in (2) for $F(\Theta_L)$.

We state a few properties of $F(\Theta_L)$ defined in (4), which explicitly identify the form of κ. All proofs of Lemmas and Theorems, as well as certain remarks, are contained in Appendix A.

Lemma 4.1. *Assume ϕ_L is K-Lipschitz continuous and monotone on its domain. For an input X,*

1. *$F(\Theta_L)$ is both monotone and K_2-Lipschitz, where $K_2 := K\mathbb{E}_X\{\|\eta(\hat{X}_L)\|_2^2$.*
2. *$\kappa = \lambda_{\min}(\nabla\phi_L)\mathbb{E}_X[\lambda_{\min}(\eta^\intercal(\hat{X}_L)\eta(\hat{X}_L))]$ if $\nabla\phi_L$ exists; it is 0 otherwise.*

For simplicity, we assume from now on the stronger condition that $\lambda_{\min}(\eta^\intercal(\hat{X}_L)\eta(\hat{X}_L)) > 0$ for any generic nonlinear feature $\hat{X}_L$. In addition, because we are considering the last layer in classification, ϕ_L is typically chosen as sigmoid or softmax, both of which are differentiable so $\lambda_{\min}(\nabla\phi_L)$ exits.

4.1 Case 1: modulus $\kappa > 0$

We first state several assumptions used in convergence analyses.

Assumption 4.2 (ϵ-approximate estimate from previous layers)**.** Let $\hat{X}_L := f^*_{1:L-1}(X)$ and $\hat{X}_L := \hat{f}_{1:L-1}(X)$ be the oracle and estimated output from the previous $L-1$ layers. There exists $\epsilon > 0$ such that $\mathbb{E}_X\|\hat{X}_L - \hat{X}_L\|_2 \leq \epsilon$.

Assumption 4.3. (1) The oracle parameter Θ_L^* of the last layer satisfies the MVI inequality (3) for F. (2) The mapping $\eta(\cdot)$ is D-Lipschitz continuous. (3) For any $\Theta_L \in \boldsymbol{\Theta}$, $\|\Theta_L\|_2 \leq B$.

We bound the recovered parameters following techniques in (Juditsky & Nemirovsky, 2019).

Lemma 4.4 (Parameter recovery guarantee)**.** *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 $\widehat{\Theta}_L^{(T)}$ obeys the error bound

$$\mathbb{E}_{\widehat{\Theta}_L^{(T)},X}\{\|\widehat{\Theta}_L^{(T)} - \Theta_L^*\|_2^2\} \leq \frac{4M^2}{\kappa^2(T+1)}. \tag{8}$$

We can use the above result to bound the error in posterior prediction.

Theorem 4.5 (Prediction error bound for model recovery, strongly monotone F)**.** *Under Assumptions 4.2 and 4.3, we have for a given test signal $X_t, t > N$ that for $p \in [2, \infty]$,*

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

where $\widehat{\mathbb{E}}[Y_t|X_t]$ 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 Theorem 4.5 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.

Remark 4.6 (When $\kappa > 0$). Recall that $\kappa = \lambda_{\min}(\nabla\phi_L)\mathbb{E}_X[\lambda_{\min}(\eta^\intercal(\hat{X}_L)\eta(\hat{X}_L))]$. We may assume that the quantity $\mathbb{E}_X[\lambda_{\min}(\eta^\intercal(\hat{X}_L)\eta(\hat{X}_L))]$ is always lower bounded away from zero, so as to only concern with $\lambda_{\min}(\nabla\phi_L)$. When ϕ_L is a point-wise function on its vector inputs, this gradient matrix is diagonal. In the case of the sigmoid function, we know that for any $y \in \mathbb{R}^n$

$$\lambda_{\min}[\nabla\phi_L]|_y = \min_{i=1,\dots,n} \phi_L(y_i)(1 - \phi_L(y_i)),$$

which is bounded away from zero. In general, we only need that the point-wise activation is continuously differentiable with positive derivatives.

4.2 Case 2: modulus $\kappa = 0$

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 $z \in \mathbb{R}^n$ (Gao & Pavel, 2018, 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) (Kotsalis et al., 2020) to obtain a similar but weaker ℓ_p prediction guarantee.

Theorem 4.7 (Prediction error bound for model recovery, monotone F)**.** *Suppose we run the OE algorithm (Kotsalis et al., 2020) 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 $\{2, 3, \dots, T\}$. Then for $p \in [2, \infty]$,*

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

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 Theorem 4.7 is also unaffected by the batch size, which only serves to reduce the variance. In addition, Theorem 4.7 requires R be uniformly chosen from $\{2, 3, \dots, T\}$, 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 ((Kotsalis et al., 2020), 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.

4.3 Equivalence between monotone operator and gradient of parameters

In practice, neural networks, including single-layer ones, are commonly trained via empirical loss minimization, in contrast to solving MVI. We now show that under the cross-entropy loss, when ϕ is either the sigmoid function or the softmax function, the monotone operator F defined in (4) coincides with the gradient of the loss with respect to parameters.

Proposition 4.8 (The equivalence between MVI and parameter gradient)**.** *Consider a generic pair of input signal X and random realization Y in binary or multi-class classification with the sigmoid or softmax function under cross-entropy loss. For any parameter Θ*

$$\mathbb{E}_{X,Y}[\nabla_{\Theta_L}\mathcal{L}(Y, \phi_L(\eta(X_L)\Theta_L))] = F(\Theta_L),$$

where the monotone operator $F(\Theta_L)$ substitutes $\eta_L(X_L)$ and Θ_L in (4).

4.4 Imprecise graph knowledge

Because graph knowledge is rarely perfectly known, the robustness of the algorithm under estimated graph topology knowledge is important in practice. Thus, we want to analyze the difference in prediction quality when the true graph adajacency matrix W is estimated by another W'. For simplicity, we focus on the GCN case and denote $\eta(\hat{X}_L)$ under true or estimated graph Laplacian as L_gX or L'_gX. We show that when L_g and L'_g are close in spectral norm, there exists a solution $\Theta^*_{L'}$ bounding the prediction error between $\mathbb{E}[Y|X]$ under Θ^*_L and $\Theta^*_{L'}$. In particular, $\Theta^*_{L'}$ is the solution of VI$[F, \mathbf{\Theta}]$ if ϕ_L is the sigmoid function so that we can guarantee a small prediction error in expectation based on earlier results in such cases. We also experimentally verify the performance of `SVI` under model mismatch in Section 5.1.

Proposition 4.9 (Quality of minimizer under model mismatch)**.** *Fix a $\delta > 0$. Assume that $\|L_g^{'} - L_g\|_2 \le \delta/[K\mathbb{E}_X\|X\|_2]$, where K is the Lipschitz continuous constant for ϕ_L. Then, there exists $\Theta^*_{L'}$ such that*

$$\|\mathbb{E}[Y|X]' - \mathbb{E}[Y|X]\|_2 \le \delta\|\Theta^*_L\|_2,$$

where $\mathbb{E}[Y|X]'$ denotes the conditional expectation under L'_g and $\Theta^*_{L'}$.

5 Experiments

We test and compare `SVI` in Algorithm 1 with widely-used stochastic gradient descent methods on various synthetic and real-data examples. In particular, we compare `SVI` or `SVI`-Adam with SGD or Adam, where SVI (resp. SVI-Adam) uses SVI in Algorithm 1 to provide update direction and optimized with SGD (resp. Adam). We demonstrate the competitive or better performance by SVI, especially in the improved efficiency in the early stage of training against the SGD baseline.

For fair experimental comparisons, all hyperparameters for SVI/SVI-Adam and SGD/Adam are tuned to be the same. Doing so is possible as SVI only provides alternative update directions during the network training, whereby the same optimization algorithm (e.g., SGD or Adam) can be subsequently used. Partial results are shown due to space limitation. Appendix B.2 describes details regarding experiment setup and comparison metrics and B.3 onward contains full results. Notation-wise, N (resp. N_1) denotes the size of training (resp. test) sample, lr denotes the learning rate, B denotes the batch size, and E denotes training epochs.

5.1 Synthetic Data Experiments

5.1.1 Two-layer FC network classification. We first classify the cluster label of the simulated two-moon dataset. Figure 8a visualizes the dataset. We use a two-layer fully-connected network with ReLU (layer 1) and softmax (layer 2) activation. We let $N = N_1 = 500$, $B = 100$, $E = 100$, lr=0.15, and momentum = 0.99. Table 1 shows that SVI consistently reaches smaller MSE losses and classification errors on training and test data across different number of hidden neurons. Figure 2 shows that SVI also converges faster than SGD throughout epochs. See Appendix B.3, Figure 8 for additional results that illustrates model intermediate convergence results.

Table 1: Two-moon data. We show average results at end of epochs, with standard errors in brackets. SVI consistently reaches smaller training and/or test MSE loss and/or classification errors than SGD across different number of hidden neurons.

Two moons	MSE loss				Classification error			
# Hidden neurons	SGD train	SVI train	SGD test	SVI test	SGD train	SVI train	SGD test	SVI test
8	0.00062 (4.0e-05)	0.00054 (2.1e-05)	0.06709 (4.8e-03)	0.05993 (4.5e-03)	0.08333 (7.2e-03)	0.076 (3.8e-03)	0.098 (6.6e-03)	0.08933 (1.1e-02)
16	0.00053 (7.7e-05)	0.00042 (8.5e-05)	0.05852 (7.1e-03)	0.04702 (7.7e-03)	0.06533 (1.5e-02)	0.052 (1.3e-02)	0.078 (9.6e-03)	0.06333 (1.3e-02)
32	0.00022 (2.9e-05)	0.00014 (1.8e-05)	0.02375 (2.2e-03)	0.01517 (6.9e-04)	0.01867 (3.8e-03)	0.01533 (2.4e-03)	0.01733 (2.2e-03)	0.016 (9.4e-04)
64	0.00016 (1.7e-05)	8e-05 (1.4e-05)	0.01774 (1.2e-03)	0.00872 (1.0e-03)	0.01 (2.5e-03)	0.006 (1.6e-03)	0.01067 (3.0e-03)	0.00933 (2.0e-03)

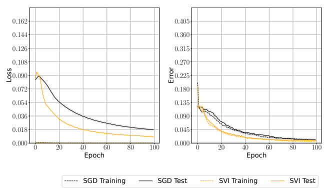


Figure 2: Two-moon data, 64 hidden neurons, with MSE loss (left) and classification error (right). SVI shows faster convergence.

5.1.2 Two-layer GCN model recovery. The underlying graphs and data-generating process are described in Appendix B.3. We aim to compare the model recovery performances measured by Eq. (16) for $p = 2, \infty$, when the true graph is known (i.e., edge sets fully known) or perturbed. We generate large random graphs and let $N = N_1 = 2000$, $B = 100$, $E = 200$, lr=0.001, and momemtum = 0.99. We also use the Nesterov momentum (Sutskever et al., 2013) when training the two-layer GCN model with ReLU activation. Table 2 shows that SVI consistently reaches smaller ℓ_2 or ℓ_∞ model recovery error and MSE error. The pattern is consistent even if the graph information is perturbed. Figure 3 shows that SVI consistently converges faster than SGD.

Table 2: Two-layer GCN model recovery on the large random graph. We observe consistently better model recovery performance by SVI, regardless of the number of hidden neurons for estimation, comparison metrics, and whether graph is perturbed. Metrics are defined in (16) and (12) respectively.

Large Graph	ℓ_2 model recovery test error				MSE test loss				ℓ_∞ model recovery test error			
# Hidden neurons	SGD (Known)	SGD (Perturbed)	SVI (Known)	SVI (Perturbed)	SGD (Known)	SGD (Perturbed)	SVI (Known)	SVI (Perturbed)	SGD (Known)	SGD (Perturbed)	SVI (Known)	SVI (Perturbed)
2	0.116 (3.4e-03)	0.117 (3.6e-03)	0.091 (8.4e-03)	0.092 (7.5e-03)	0.25 (1.9e-04)	0.25 (2.0e-04)	0.249 (4.2e-04)	0.249 (3.6e-04)	0.129 (4.3e-03)	0.13 (4.5e-03)	0.1 (8.6e-03)	0.102 (7.3e-03)
4	0.1 (8.1e-03)	0.101 (7.6e-03)	0.078 (6.2e-03)	0.081 (5.6e-03)	0.25 (4.0e-04)	0.25 (3.7e-04)	0.248 (3.0e-04)	0.249 (2.8e-04)	0.109 (8.4e-03)	0.11 (7.8e-03)	0.087 (4.8e-03)	0.091 (3.8e-03)
8	0.084 (7.7e-03)	0.086 (7.0e-03)	0.067 (5.5e-04)	0.07 (4.4e-04)	0.249 (3.4e-04)	0.249 (3.2e-04)	0.248 (4.0e-06)	0.248 (1.0e-06)	0.088 (7.2e-03)	0.091 (6.5e-03)	0.076 (8.2e-04)	0.083 (5.2e-04)
16	0.08 (7.3e-03)	0.081 (6.5e-03)	0.066 (3.3e-04)	0.07 (2.1e-04)	0.249 (2.8e-04)	0.249 (2.5e-04)	0.248 (6.0e-06)	0.248 (1.0e-05)	0.087 (7.8e-03)	0.089 (6.6e-03)	0.075 (4.6e-04)	0.082 (1.4e-04)
32	0.082 (8.1e-03)	0.084 (7.5e-03)	0.066 (5.6e-05)	0.07 (6.2e-05)	0.249 (3.6e-04)	0.249 (3.4e-04)	0.248 (1.1e-05)	0.248 (1.2e-05)	0.089 (8.8e-03)	0.091 (7.9e-03)	0.075 (1.7e-04)	0.082 (1.0e-04)

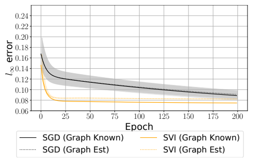
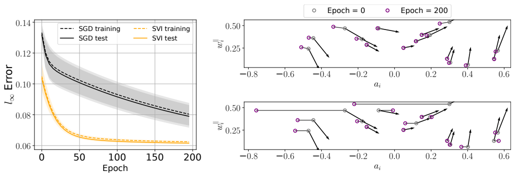
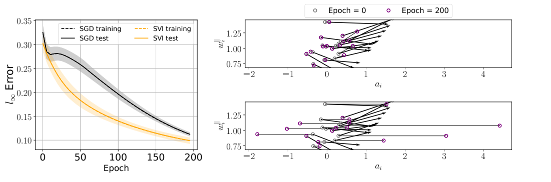


Figure 3: ℓ_∞ model recovery error on the large random graph. SVI reaches smaller error under faster initial convergence.



(a) Two-layer GCN



(b) Three-layer GCN

Figure 4: ℓ_∞ model recovery error with 16 neurons. Left: relative l_∞ error assuming the true graph is known. Right: visualization of the training dynamics by SGD (top) and SVI (bottom). After 200 epochs, SVI displaces the neurons from their initial position further and leads to faster convergence.

5.2 Real-data Experiments

5.2.1 Multi-class traffic flow anomaly detection. The data collection and description are described in Appendix B.4. The goal is identify anomalous bi-hourly traffic flow observations. We train three-layer GCN models with lr = 0.001, momemtum = 0.99, and the Nesterov momentum. Table 3 shows that SVI reaches smaller training and test classification error for large hidden neurons and remains competitive when fewer neurons are used. In terms of weighted F_1 scores, SVI also reaches higher training and test scores in nearly all cases. Figure 5 shows that SVI converges faster than SGD, which aligns with simulation results. See Appendix B.4, Figure 12 for intermediate convergence results.

Table 3: Traffic data multi-class anomaly detection. We note that SVI remains competitive or outperforms SGD in terms of classification error and weighted F_1 scores, with clear improvement when hidden neurons increase.

Traffic data	Cross-Entropy loss				Classification error				Weighted F_1 score			
# Hidden neurons	SGD Training	SGD Test	SVI Training	SVI Test	SGD Training	SGD Test	SVI Training	SVI Test	SGD Training	SGD Test	SVI Training	SVI Test
8	0.837 (7.0e-03)	0.834 (1.1e-02)	0.967 (6.2e-03)	0.969 (5.5e-03)	0.397 (1.3e-02)	0.414 (1.5e-02)	0.412 (1.6e-02)	0.427 (1.6e-02)	0.572 (2.9e-02)	0.554 (2.9e-02)	0.572 (2.4e-02)	0.558 (2.3e-02)
16	0.808 (5.6e-03)	0.804 (4.1e-03)	0.955 (3.3e-03)	0.956 (4.0e-03)	0.391 (1.3e-02)	0.411 (1.2e-02)	0.37 (3.8e-03)	0.384 (7.0e-03)	0.576 (3.1e-02)	0.556 (3.0e-02)	0.624 (7.3e-03)	0.61 (1.0e-02)
32	0.791 (5.7e-03)	0.787 (6.5e-03)	0.939 (3.4e-03)	0.941 (3.6e-03)	0.366 (5.9e-03)	0.383 (7.9e-03)	0.358 (4.5e-03)	0.37 (4.9e-03)	0.628 (6.7e-03)	0.611 (9.0e-03)	0.641 (4.6e-03)	0.629 (5.0e-03)
64	0.791 (2.0e-03)	0.787 (2.2e-03)	0.933 (2.1e-03)	0.935 (2.1e-03)	0.36 (2.7e-03)	0.375 (4.4e-03)	0.346 (1.7e-03)	0.358 (1.6e-03)	0.637 (2.8e-03)	0.622 (4.7e-03)	0.652 (2.0e-03)	0.641 (1.8e-03)

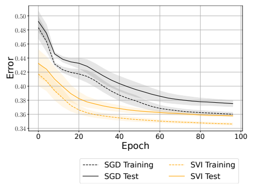


Figure 5: Classification error under 64 neurons. SVI converges faster than SGD.

5.2.2 Vision datasets. We compare SVI with gradient-based method on training FC networks (FCNet) and LeNet (LeCun et al., 1998). The FCNet has four layers with 512 hidden nodes in each hidden layer. We examine performance on the MNIST dataset (LeCun, 1998) and the CIFAR-10 dataset (Krizhevsky et al., 2009), where the lr is fixed as 0.001 throughout training. We train the network for 10 epochs and fix batch size as 64, so each epoch contains 938 (resp. 782) effective iterations (i.e., mini-batches) on MNIST (resp. CIFAR-10). To stabilize the performance of SVI in later iterations, we sometimes use a warm-start technique—the estimates by SVI after the initial 20% iterations initialize parameters for gradient-based methods in the rest 80% iterations. We do so because SVI shows faster initial convergence but due to the skipping step, may lowers final accuracy. Table 4 shows results under both networks, where SVI-SGD outperforms SGD on both data and networks. SVI-Adam is competitive against Adam but the benefits are unclear; we expect this situation due to a lack of theoretical analyses (even for the last layer) under Adam, which belong to future work. Nevertheless, the highest test accuracy by SVI upon fixing the dataset and network is competitive/higher than that by gradient-based methods. Figure 6 further visualizes training and test accuracies along the entire training iterations and shows faster initial convergence by SVI.

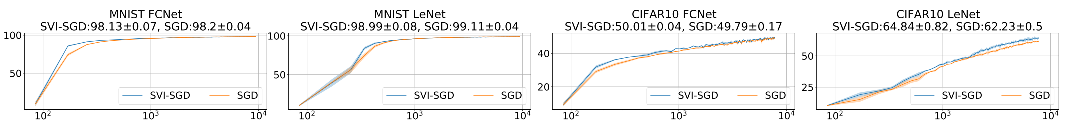


Figure 6: Classification test accuracies on both datasets under two network architecture. SVI-SGD improves initial convergence over SGD and yields competitive/better final performance.

Table 4: Classification accuracy of `SVI` against gradient-based method on two vision datasets under two separate networks. *full* indicates results after all training iterations, and *initial* indicates results after 5% of the total iterations. Entries in brackets indicates standard deviation over 3 independent initialization of model parameters.

	SVI-SGD		SGD		SVI-Adam		Adam	
FCNet	Train	Test	Train	Test	Train	Test	Train	Test
MNIST full	99.1 (0.1)	98.1 (0.1)	99.1 (0.1)	98.2 (0.0)	97.7 (0.1)	97.5 (0.1)	97.9 (0.1)	97.5 (0.0)
MNIST initial	93.5 (0.1)	93.7 (0.1)	92.3 (0.2)	92.6 (0.2)	92.6 (0.3)	93.0 (0.2)	92.6 (0.3)	92.9 (0.2)
CIFAR-10 full	55.8 (0.8)	50.0 (0.0)	55.5 (0.7)	49.8 (0.2)	48.8 (1.2)	46.5 (0.6)	48.4 (0.6)	45.8 (0.2)
CIFAR-10 initial	36.9 (0.6)	37.7 (0.1)	34.6 (0.4)	35.5 (0.6)	34.5 (0.9)	34.6 (0.7)	33.2 (1.2)	33.5 (1.8)
	SVI-SGD		SGD		SVI-Adam		Adam	
LeNet	Train	Test	Train	Test	Train	Test	Train	Test
MNIST full	99.0 (0.1)	99.0 (0.1)	99.0 (0.1)	99.1 (0.0)	99.6 (0.1)	99.2 (0.0)	99.7 (0.0)	99.3 (0.0)
MNIST initial	90.2 (1.1)	90.4 (1.1)	86.0 (2.0)	86.6 (1.8)	96.3 (0.5)	96.7 (0.4)	96.8 (0.6)	97.3 (0.4)
CIFAR-10 full	71.7 (0.8)	64.8 (0.8)	63.9 (0.4)	62.2 (0.5)	70.3 (0.8)	63.7 (0.8)	71.8 (0.5)	64.8 (0.6)
CIFAR-10 initial	23.7 (1.1)	24.6 (1.1)	22.8 (0.7)	23.4 (0.5)	41.3 (0.6)	41.7 (0.7)	43.1 (0.5)	43.5 (0.2)

5.2.3 Multi-class large-scale OGB node classification. We lastly demonstrate the applicability of `SVI` on the large `ogbn-arxiv` graph provided by the Open Graph Benchmark (Hu et al., 2020; 2021). The graph is much larger than earlier examples, and we also use wider and deeper models; more details are in Appendix B.5. We optimize with `SVI`, `SVI`-Adam, SGD, and Adam, and train for a fixed $E = 500$ epochs. Figure 7 shows that `SVI` or `SVI`-Adam yield comparable results to SGD or Adam. In addition, the bottom row shows clearly that `SVI` and `SVI`-Adam converge much faster than SGD and Adam during initial epochs. We believe this faster initial convergence is particularly useful on large-scale experiments, where it is computationally demanding to train networks. See Appendix B.5 for results under different number of hidden neurons (cf. Figure 13) and under different choices of learning rate (cf. Table 6).

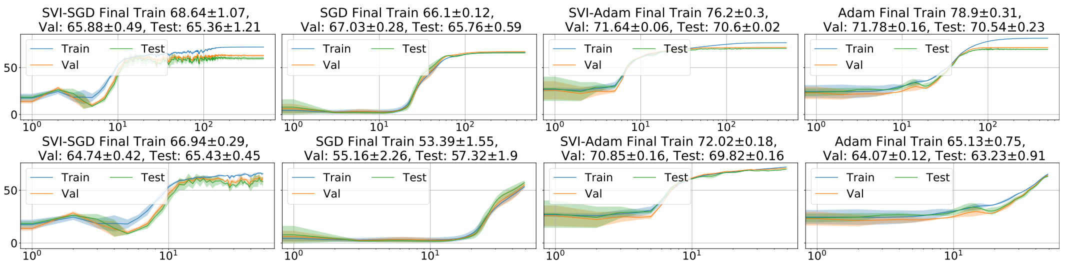


Figure 7: Classification accuracies on the large-scale `ogbn-arxiv` dataset. The top (resp. bottom) row shows accuracies over all (resp. initial 50) epochs, where "final" indicates results at highest validation accuracies. `SVI`-based methods improve initial convergence with competitive final accuracies.

6 Conclusion

We have investigated how a new MVI approach can be useful in training neural networks, with strong prediction guarantees in special cases and competitive performance as SGD or Adam. In essence, the algorithm `SVI` "skips" computing derivative of the point-wise non-linearities. Although this skipping seems counter-intuitive at first glance, the theoretical justifications in the idealized theoretical settings of training the last-layer, assuming previous layers are fully known, provides partial justifications to the procedure. On various synthetic and real-data examples, our `SVI` seems to improve efficiency in the early stage of training, which is useful when computational resources are constrained.

At present, the following tasks remain open. Practically, testing the performance of `SVI` on training large models for more complex problems is worth studying. Theoretically, the guarantees only hold when all except the last layers of the network are known. Instead, it is important to relax and quantify the level of knowledge we need for these layers to better explain the previous-layer heuristics of `SVI`. Application-wise, it is useful to address a wider range of problems in GNN, including edge and graph classification (Zhou et al., 2020). We will explore them in the future.

REFERENCES

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

Ehsan Amid, Rohan Anil, and Manfred Warmuth. Locoprop: Enhancing backprop via local loss optimization. In *International Conference on Artificial Intelligence and Statistics*, pp. 9626–9642. PMLR, 2022.

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*, pp. 8139–8148, 2019a.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. *arXiv preprint arXiv:1901.08584*, 2019b.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29:3844–3852, 2016.

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*, pp. 1675–1685. PMLR, 2019.

Francisco Facchinei and J. S. Pang. Finite-dimensional variational inequalities and complementarity problems. 2003.

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

Bolin Gao and Lacra Pavel. On the properties of the softmax function with application in game theory and reinforcement learning, 2018.

William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 1025–1035, 2017.

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. *arXiv preprint arXiv:2005.00687*, 2020.

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.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ArXiv*, abs/1502.03167, 2015.

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

David Kinderlehrer and Guido Stampacchia. An introduction to variational inequalities and their applications. 1980.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Georgios Kotsalis, Guanghui Lan, and Tianjiao Li. Simple and optimal methods for stochastic variational inequalities, i: operator extrapolation. *arXiv preprint arXiv:2011.02987*, 2020.

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

Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

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.

Qihang Lin, Mingrui Liu, Hassan Rafique, and Tianbao Yang. Solving weakly-convex-weakly-concave saddle-point problems as weakly-monotone variational inequality. 2018.

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.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv preprint arXiv:1412.6614*, 2014.

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.

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. 2017.

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 (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Mert Pilanci and Tolga Ergen. Neural networks are convex regularizers: Exact polynomial-time convex optimization formulations for two-layer networks. In *ICML*, 2020.

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 (eds.), *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1139–1147, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32:4–24, 2019.

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.

Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Graph neural networks: A review of methods and applications. *ArXiv*, abs/1812.08434, 2020.

A PROOFS

Proof of Lemma 4.1. We first verify the monotonicity of F by considering vectorized parameters, where for a matrix $A \in \mathbb{R}^{m\times n}$, $\text{vec}(A) \in \mathbb{R}^{mn}$ by stacking vertically columns of A. Note that this vectorization is simply used to make sure the Euclidean inner product between F and Θ_L is well-defined; the fundamental meaning of Θ_L (e.g., as the channel-mixing coefficient) remains unchanged. With an abuse of notation, we use the same Θ_L and $\boldsymbol{\Theta}$ to denote the vectorized parameter and the corresponding parameter space. For any $\Theta_{1,L}, \Theta_{2,L} \in \boldsymbol{\Theta}$,

$$\begin{aligned}
&\langle F(\Theta_{1,L}) - F(\Theta_{1,L}), \Theta_{1,L} - \Theta_{2,L}\rangle \\
&= \langle \mathbb{E}_X\{\eta^\intercal(\hat{X}_L)(\phi_L(\eta(\hat{X}_L)\Theta_{1,L}) - \phi_L(\eta(\hat{X}_L)\Theta_{2,L})\}, \Theta_{1,L} - \Theta_{2,L}\rangle \\
&= \mathbb{E}_X\{(\phi_L(\eta(\hat{X}_L)\Theta_{1,L}) - \phi_L(\eta(\hat{X}_L)\Theta_{2,L}))^T(\eta(\hat{X}_L)\Theta_{1,L} - \eta(\hat{X}_L)\Theta_{2,L})\} \\
&\geq \lambda_{\min}(\nabla\phi_L)\mathbb{E}_X\{\|\eta(\hat{X}_L)\Theta_{1,L} - \eta(\hat{X}_L)\Theta_{2,L}\|_2^2\} \qquad \text{(If } \nabla\phi_L \text{ exists.)} \\
&\geq \underbrace{\lambda_{\min}(\nabla\phi_L)\mathbb{E}_X\{\lambda_{\min}(\eta^\intercal(\hat{X}_L)\eta(\hat{X}_L))}_{\kappa:=}\|\Theta_{1,L} - \Theta_{2,L}\|_2^2\},
\end{aligned}$$

where κ is defined in (2). The first equality uses the fact that the Y part is cancelled and the first inequality holds when ϕ_L is *continuously differentiable* on its domain. If ϕ_L is only monotone on its domain, the the first inequality will only be greater than 0.

We then verify the K_2-Lipschitz continuity of F. For any $\Theta_{1,L}, \Theta_{2,L} \in \boldsymbol{\Theta}$,

$$\begin{aligned}
\|F(\Theta_{1,L}) - F(\Theta_{2,L})\|_2 &= \mathbb{E}_X\{\|\eta^\intercal(\hat{X}_L)(\phi_L(\eta(\hat{X}_L)\Theta_{1,L}) - \phi_L(\eta(\hat{X}_L)\Theta_{2,L})\|_2\} \\
&\leq \mathbb{E}_X\{\|\eta^\intercal(\hat{X}_L)\|_2\|\phi_L(\eta(\hat{X}_L)\Theta_{1,L}) - \phi_L(\eta(\hat{X}_L)\Theta_{2,L})\|_2\} \\
&\leq K\mathbb{E}_X\{\|\eta(\hat{X}_L)\|_2\|\eta(\hat{X}_L)\Theta_{1,L} - \eta(\hat{X}_L)\Theta_{2,L}\|_2\} \\
&\leq \underbrace{K\mathbb{E}_X\{\|\eta(\hat{X}_L)\|_2^2\}}_{K_2:=}\|\Theta_{1,L} - \Theta_{2,L}\|_2^2,
\end{aligned}$$

where K_2 has been defined. We repeated used the Cauchy–Schwarz inequality and the last inequality relies on the assumption that ϕ_L is K-Lipschitz continuous.

Note that given random samples $\{X_1, \ldots, X_N\}$, the quantities $\mathbb{E}_X\{\lambda_{\min}(\eta^\intercal(\hat{X}_L)\eta(\hat{X}_L))\}$ and $\mathbb{E}_X\{\|\eta(\hat{X}_L)\|_2^2\}$ can be empirically approximated by sample averages.

□

Remark A.1 (Remark for Lemma 4.1). For simplicity, we assume from now on the stronger condition that $\lambda_{\min}(\eta^\intercal(\hat{X}_L)\eta(\hat{X}_L)) > 0$ for any generic nonlinear feature $\hat{X}_L$. In addition, because we are considering the last layer in classification, ϕ_L is typically chosen as sigmoid or softmax, both of which are differentiable so $\lambda_{\min}(\nabla\phi_L)$ exits.

Proof of Lemma 4.4. The proof employs classical techniques when analyzing the convergence of projection descent in stochastic optimization, which appear in (Juditsky & Nemirovsky, 2019, Proposition 3.2).

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 4.3 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 $\mathbf{\Theta}$ (e.g., $\|\Theta_1 - \Theta_2\|_2^2 \leq B^2\ \forall(\Theta_1,\Theta_2) \in \mathbf{\Theta}$. 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 of Theorem 4.5. Define

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

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

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

We now bound (a) and (b) separately.

Bound of (a). We have that

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

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 4.2 and 4.3 that

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

Proof of Theorem 4.7. The crux is to bound the expected value of the norm of F evaluated at the stochastic OE estimate. This bound results from (Kotsalis et al., 2020, Theorem 3.8), where in general, for any $\Theta_L \in \mathbf{\Theta}$, the authors use the residual

$$\mathbb{E}[\text{res}(\Theta_L)] \le \delta,\ \text{res}(\Theta_L) := \min_{y\in -N_{\mathbf{\Theta}}(\Theta_L)} \|y-F(\Theta_L)\|_2$$

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 $N_{\mathbf{\Theta}}(\Theta_L) := \{y\in\mathbb{R}^p|\langle y,\Theta'-\Theta_L\rangle,\forall\Theta'\in\mathbf{\Theta}\}$ denotes the normal cone of $\mathbf{\Theta}$ at Θ_L. Then, under the assumptions on F and choices of step sizes, we can restate (Kotsalis et al., 2020, Theorem 3.8) in our special case as

$$\mathbb{E}_{\hat{\Theta}_L^{(R)}}[\text{res}(\hat{\Theta}_L^{(R)})] \le \frac{3\sigma}{\sqrt{T}} + \frac{12K_2\sqrt{2\|\Theta_L^*\|_2^2+\frac{2\sigma^2}{L^2}}}{\sqrt{T}}.$$

When we assume $\mathbf{\Theta}$ is the entire space, $N_{\mathbf{\Theta}}(\Theta_L)=\{\mathbf{0}\}$ whereby $\text{res}(\hat{\Theta}_L^{(R)})=\|F(\hat{\Theta}_L^{(R)})\|_2$. Thus, the result follows.

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

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

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

$$\mathbb{E}_{\hat{\Theta}_L^{(R)}}\{\|\mathbb{E}_X\{\sigma_{\min}(\eta(\hat{X}_L))[\widehat{\mathbb{E}}[Y_t|X_t]-\mathbb{E}[Y_t|X_t]\}\|_p\} \le \mathbb{E}_{\hat{\Theta}_L^{(R)}}\|F(\hat{\Theta}_L^{(R)})\|_2,$$

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

Proof of Proposition 4.8. For notation simplicity, denote $\eta := \eta_L(X_L)$ and $\theta := \Theta_L$. Meanwhile, for two vectors $a,b\in\mathbb{R}^n$, the notation a/b denotes the point-wise division. $\mathbf{1}$ denotes a vector of all 1. In addition, the binary and categorical cross-entropy losses are defined as:

$$\mathcal{L}(Y,\phi_L(\eta(X_L)\Theta_L)) := -Y\ln(\phi(\eta_L(X_L)\Theta_L)) - (1-Y)\ln(1-\phi(\eta_L(X_L)\Theta_L)) \quad Y\in\{0,1\}. \tag{10}$$

$$\mathcal{L}(Y,\phi_L(\eta(X_L)\Theta_L)) := -e_Y^T\ln(\phi(\eta_L(X_L)\Theta_L)) \qquad Y\in\{0,\dots,F\}, F>1. \tag{11}$$

In (10), $\phi(x)=\exp(x)/(1+\exp\{x\})$ is the sigmoid function applied point-wise. In (11), e_k is the k^{th} standard basis vector in $\mathbb{R}^{F+1}$ and $\phi(\boldsymbol{x})=\exp(\boldsymbol{x}_i)/\sum_j\exp(\boldsymbol{x}_j)$ is the softmax function applied row-wise.

We first consider the binary cross-entropy loss $\mathcal{L}(Y,\phi_L(\eta(X_L)\Theta_L))$ defined in (10). Note that we have

$$\begin{aligned}
\mathcal{L}(Y,\phi_L(\eta(X_L)\Theta_L)) &= -Y\ln(\phi(\eta_L(X_L)\Theta_L)) - (1-Y)\ln(1-\phi(\eta_L(X_L)\Theta_L))\\
&= -Y^T\ln(\exp(\eta\theta)/(\mathbf{1}+\exp(\eta\theta))) - (\mathbf{1}-Y)^T\ln(\mathbf{1}/(\mathbf{1}+\exp(\eta\theta)))\\
&= -Y^T\eta\theta + Y^T\ln(\mathbf{1}+\exp(\eta\theta)) + (\mathbf{1}-Y)^T\ln(\mathbf{1}+\exp(\eta\theta))\\
&= \mathbf{1}^T\ln(\mathbf{1}+\exp(\eta\theta)) - Y^T\eta\theta.
\end{aligned}$$

Thus, the gradient with respect to θ (which is Θ_L) can be written as

$$\nabla_\theta \mathcal{L}(Y, \phi_L(\eta(X_L)\Theta_L)) = \eta^T \frac{\exp(\eta\theta)}{\mathbf{1}+\exp(\eta\theta)} - \eta^T Y = \eta_L^\intercal(X_L)[\phi(\eta_L(X_L)\Theta_L) - Y].$$

Taking expectation thus yields the result.

We next consider the categorical cross-entropy loss $\mathcal{L}(Y, \phi_L(\eta(X_L)\Theta_L))$ defined in (11). Note that we have

$$\begin{aligned}\mathcal{L}(Y, \phi_L(\eta(X_L)\Theta_L)) &= -e_Y^T \ln(\phi(\eta_L(X_L)\Theta_L)) \\ &= \left[-e_Y^T \ln\left(\frac{\exp(\eta\theta)}{\mathbf{1}^T \exp(\eta\theta)}\right)\right] \\ &= -e_Y^T(\eta\theta) + \ln\big(\mathbf{1}^T \exp(\eta\theta)\big).\end{aligned}$$

Thus, the gradient with respect to θ can be written as

$$\begin{aligned}\nabla_\theta \mathcal{L}(Y, \phi_L(\eta(X_L)\Theta_L)) &= -\eta^T e_Y + \eta^T \frac{\exp(\eta\theta)}{\mathbf{1}^T \exp(\eta\theta)} \\ &= \eta^T[\phi(\eta\theta) - e_Y] = \eta_L^\intercal(X_L)[\phi(\eta_L(X_L)\Theta_L) - Y],\end{aligned}$$

where in the definition of $F(\Theta)$ in (4), $Y = e_Y \in \mathbb{R}^{F+1}$ if Y belongs to more than 2 classes. Taking expectation thus yields the result. □

Proof of Proposition 4.9. Note that we are equivalently showing that there exists $\Theta_{L'}^*$ such that

$$\|\mathbb{E}_X[\phi(L_g' X\Theta_{L'}^*) - \phi(L_g X\Theta_L^*)]\|_2 \leq \delta\|\Theta_L^*\|_2.$$

Because $\Theta_{L'}^*$ is the minimizer of ℓ_2 error, it is clear that

$$\mathbb{E}_X[\phi(L_g' X\Theta_{L'}^*) - \phi(L_g X\Theta_L^*)]\|_2 \leq \mathbb{E}_X[\phi(L_g' X\Theta_L^*) - \phi(L_g X\Theta_L^*)]\|_2.$$

Now,

$$\begin{aligned}\|\mathbb{E}_X[\phi(L_g' X\Theta_L^*) - \phi(L_g X\Theta_L^*)]\|_2 &\leq \mathbb{E}_X\|[\phi(L_g' X\Theta_L^*) - \phi(L_g X\Theta_L^*)]\|_2 \\ &\leq K\mathbb{E}_X\|[L_g' X\Theta_L^* - L_g X\Theta_L^*]\|_2 \\ &\leq K\mathbb{E}_X\|(L_g' - L_g)X\|_2\|\Theta_L^*\|_2. \\ &\leq K\|L_g - L_g^{'}\|_2\mathbb{E}_X\|X\|_2\|\Theta_L^*\|_2 \\ &\leq \delta\|\Theta_L^*\|_2.\end{aligned}$$

Therefore, the loss under the true parameter Θ_L^* is bounded above by $\delta\|\Theta_L^*\|_2$, so there exists a $\Theta_{L'}^*$ that achieves no larger error. □

B ADDITIONAL DETAILS

Setup and comparison metrics are described in Appendix B.2. We describe dataset details and show additional results in Appendices B.3 onward, which illustrate the behavior of SVI and SGD during the entire training epoch.

B.1 GNN NOTATION

Our notation of GNN models (i.e., graph filtering) is standard. Suppose we have an undirected and connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$, where $\mathcal{V}$ is a finite set of n vertices, $\mathcal{E}$ is a set of edges, and $W \in \mathbb{R}^{n\times n}$ is a weighted adjacency matrix that encodes node connections. Let I_n denote an identity matrix of size n. Let D be the degree matrix of W and $L_g = I_n - D^{-1/2}WD^{-1/2}$ be the normalized graph Laplacian, which has the eigen-decomposition $L_g = U\Lambda U^T$. For a graph signal $X \in \mathbb{R}^{n\times C}$ with C input channels, it is then filtered via a function $g_\Theta(L_g)$ which acts on L_g with channel-mixing parameters $\Theta \in \mathbb{R}^{C\times F}$ for F output channels. Thus, the filtered signal $X' = g_\Theta(L_g)X$.

B.2 Setup and Comparison Metrics

Setup. All implementation are done using `PyTorch` (Paszke et al., 2019) and `PyTorch Geometric` (Fey & Lenssen, 2019) (for GNN). Models are trained with one Tesla P100-PCIE-16GB. Total amount of computing time is not reported, as `SVI` only alters the update direction, which is identical to gradient-descent methods in special cases (see Proposition 4.8), so that `SVI` requires nearly identical computing resources as SGD. To ensure fair comparison, we carefully describe the experiment setup. In particular, the following inputs are *identical* to both `SVI` and SGD in 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.

Training regime: (a) parameter initialization (b) hyperparameters for optimizers (e.g., learning rate, momemtum factor, acceleration) (c) total number of epochs

In short, all except the way gradients are defined are kept the same for each comparison—our proposed `SVI` backpropagates gradients with respect to hidden input and transform the gradient back to parameter domain, whereas SGD do so with respect to parameters in each hidden layer.

Comparison metrics. For a random feature $X \in \mathbb{R}^{n\times C}$, where n is the number of graph nodes and C is the input dimension, let the true (or predicted) model be $\mathbb{E}[Y|X,\Theta] \in \mathbb{R}^{n\times F}$ (or $\mathbb{E}[Y|X,\widehat{\Theta}]$), where F is the output dimension. Given N realized pairs $\{(X_i,Y_i)\}_{i=1}^N$ where N denotes either the training or test sample size, we employ the following metrics for various tasks.[3]

$$\text{MSE loss} := N^{-1}\sum_{i=1}^N\sum_{j=1}^n \|\mathbb{E}[Y_i|X_i,\widehat{\Theta}]_j - Y_{i,j}\|_2 \tag{12}$$

$$\text{Cross-entropy loss} := N^{-1}\sum_{i=1}^N\sum_{j=1}^n Y_{i,j}^T\mathbb{E}[Y_i|X_i,\widehat{\Theta}]_j \tag{13}$$

$$\text{Classification error} := (n\cdot N)^{-1}\sum_{i=1}^N\sum_{j=1}^n\sum_{f=1}^F \mathbf{1}(Y_{i,j,f} \neq \hat{Y}_{i,j,f}) \tag{14}$$

$$\ell_p \text{ parameter recovery error} := \|\hat{\Theta} - \Theta\|_p \tag{15}$$

$$\ell_p \text{ model recovery error} := N^{-1}\sum_{i=1}^N\sum_{j=1}^n \|\mathbb{E}[Y_i|X_i,\widehat{\Theta}]_j - \mathbb{E}[Y_i|X_i,\Theta]_j\|_p. \tag{16}$$

For GCN model recovery We let $p = 2$ or ∞ in (15) and (16) and when $p = 2$, compute the relative error using $\|\Theta\|_p$ or $N^{-1}\sum_{i=1}^N\sum_{j=1}^n \|\mathbb{E}[Y_i|X_i,\Theta]_j\|_p$ on the denominator. In addition, all results are averaged over three random trials, in which features X_i are redrawn for simulated example and networks re-initialized. We show standard errors in tables as brackets and in plots as error bars. Partial results are shown due to space limitation, where Appendix B contains the extensive results.

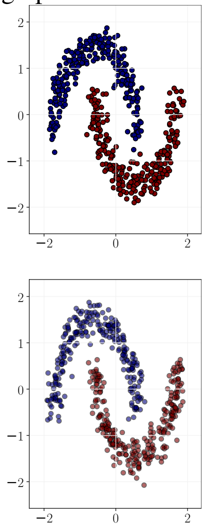
B.3 Two-layer FC networks and GCN

1. Graph data generation. Given a graph $G = (\mathcal{V},\mathcal{E})$, $|\mathcal{V}| = n$ and a signal $X_i \in \mathbb{R}^{n\times C}$, we generate $Y_i \in \mathbb{R}^{n\times F}$, where $\mathbb{E}[Y_i|X_i]$ is a two/three-layer GCN model with ReLU (layer 1/layer 1 and 2) and sigmoid (layer 2/layer 3) activation. We let $C = 2$ and $F = 1$ and let the true number of hidden nodes in each hidden layer always be 2. Entries of all true weight and bias parameters (resp. features X_i) are i.i.d. samples from $N(1,1)$ (resp. $N(0,1)$) under a fixed seed (resp. fixed seeds per random trial). We consider both small ($n = 15$) and large ($n = 40$) graphs, where $\mathbb{P}[(i,j) \in \mathcal{E}] = 0.15$; Figure 9 visualizes the graphs. The true parameters are identical in both graphs.

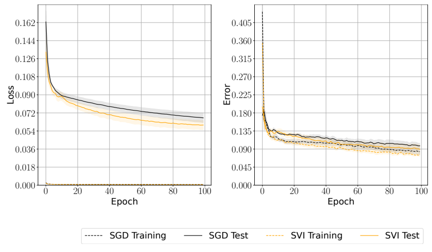
To perturb the true graph, we perturb the edge set—a p fraction of edges in $\mathcal{E}$ and in $\mathcal{E}^C$ are randomly discarded and inserted, where $\mathcal{E}^C$ denotes edges in a fully-connected graph that does not exist in $\mathcal{E}$. We set $p = 0.2$ (resp. 0.05) for small (resp. large) graphs.

[3] We one-hot encode Y_i in multi-class classification to make sure certain operations are well-defined.

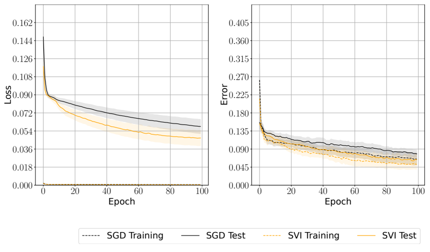
2. *Summary.* We briefly explain what each figure or table in this section contains. Figure 8 shows model intermediate convergence results for the FC networks. Figure 9 visualizes the small and large random graphs. Table 5 and Figure 10 shows additional results regarding model recovery on the small random graph. Figure 11 shows intermediate convergence results on both small and large random graphs.



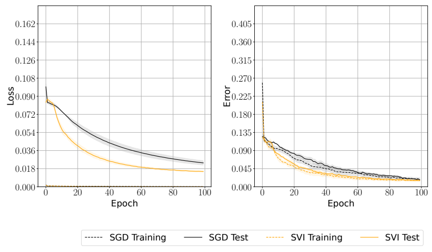
(a) Top (bottom) shows training (test) data.



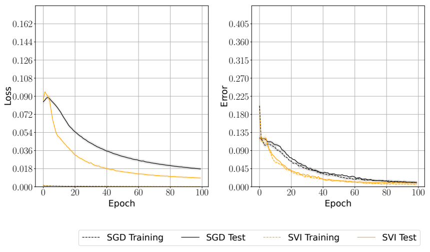
(b) 8 Hidden neurons. Left to right: MSE loss and classification error.



(c) 16 Hidden neurons. Left to right: MSE loss and classification error.

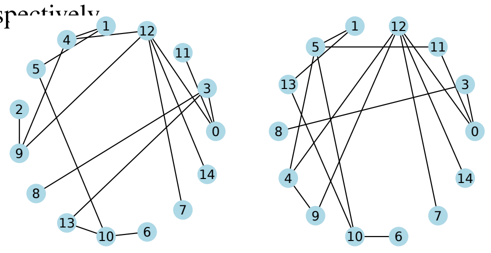


(d) 32 Hidden neurons. Left to right: MSE loss and classification error.

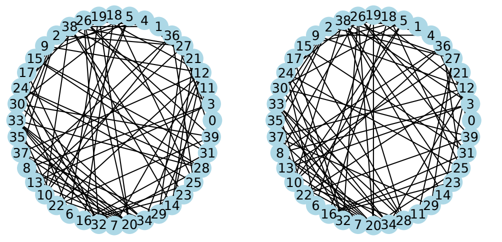


(e) 64 Hidden neurons. Left to right: MSE loss and classification error.

Figure 8: Two-moon FC network. Results are plotted with one standard error bars. In particular, SVI exhibits faster convergence than SGD and shows smaller errors. Metrics are defined in (12) and (14) respectively.



(a) Small graph, ground truth (left) and estimated graph (right)



(b) Large graph, ground truth (left) and estimated graph (right)

Figure 9: Illustration of small and random graphs.

Table 5: Two-layer GCN model recovery on the small random graph, with identical learning rate and ReLU activation for both SVI and SGD under $B = 100$. We measure recovery performance according to relative errors in predicting the posterior probability $\mathbb{E}[Y_i|X_i]$ on test data. The middle column shows the MSE loss on test data. We observe consistently better model recovery performance by SVI, regardless of the number of hidden neurons for estimation, comparison metrics, and whether graph is perturbed. Metrics are defined in (16) and (12) respectively.

Small Graph	ℓ_2 model recovery test error				MSE test loss				ℓ_∞ model recovery test error			
# Hidden neurons	SGD (Known)	SGD (Perturbed)	SVI (Known)	SVI (Perturbed)	SGD (Known)	SGD (Perturbed)	SVI (Known)	SVI (Perturbed)	SGD (Known)	SGD (Perturbed)	SVI (Known)	SVI (Perturbed)
2	0.123 (3.3e-03)	0.125 (4.9e-03)	0.094 (9.7e-03)	0.103 (5.4e-03)	0.25 (2.3e-04)	0.25 (3.0e-04)	0.249 (4.4e-04)	0.249 (3.0e-04)	0.115 (2.9e-03)	0.12 (4.9e-03)	0.089 (8.9e-03)	0.1 (3.9e-03)
4	0.105 (8.7e-03)	0.113 (5.5e-03)	0.081 (5.8e-03)	0.096 (2.8e-03)	0.249 (3.9e-04)	0.25 (2.8e-04)	0.248 (2.1e-04)	0.249 (1.4e-04)	0.099 (7.9e-03)	0.107 (4.4e-03)	0.077 (5.3e-03)	0.095 (1.8e-03)
8	0.087 (8.0e-03)	0.102 (5.8e-03)	0.067 (5.6e-04)	0.088 (1.3e-04)	0.248 (3.2e-04)	0.249 (2.6e-04)	0.248 (2.1e-05)	0.249 (2.3e-05)	0.081 (7.2e-03)	0.097 (4.7e-03)	0.064 (6.4e-04)	0.09 (1.6e-04)
16	0.084 (7.7e-03)	0.098 (4.6e-03)	0.065 (4.0e-04)	0.088 (1.5e-04)	0.248 (2.9e-04)	0.249 (2.1e-04)	0.248 (7.0e-06)	0.248 (2.5e-05)	0.079 (7.1e-03)	0.094 (3.4e-03)	0.062 (3.3e-04)	0.09 (1.3e-04)
32	0.085 (9.2e-03)	0.1 (6.2e-03)	0.065 (4.3e-04)	0.088 (3.6e-04)	0.248 (4.1e-04)	0.249 (3.3e-04)	0.248 (2.3e-05)	0.248 (3.8e-05)	0.081 (8.4e-03)	0.096 (4.8e-03)	0.062 (5.0e-04)	0.09 (1.3e-04)

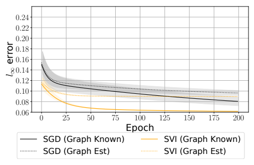
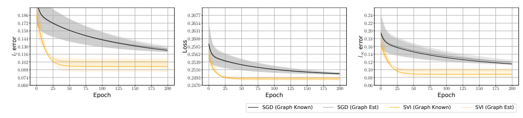
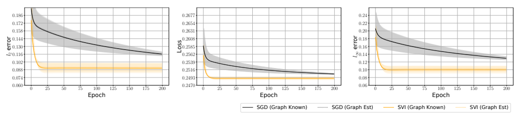


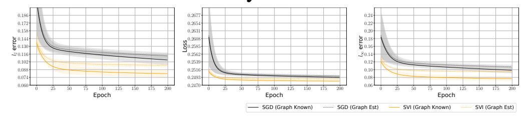
Figure 10: ℓ_∞ model recovery error on the small random graph. SVI consistently reaches smaller error under faster convergence, even just after the first training epoch.



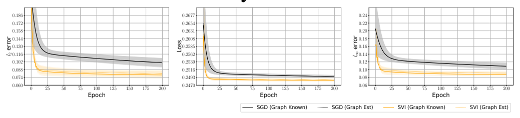
(a) Small graph, 2 hidden neurons. Left to right: ℓ_2 model recovery test error, MSE test loss, ℓ_∞ model recovery test error.



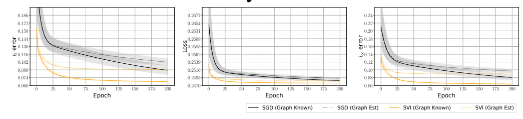
(b) Large graph, 2 hidden neurons. Left to right: ℓ_2 model recovery test error, MSE test loss, ℓ_∞ model recovery test error.



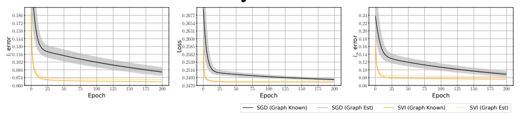
(c) Small graph, 4 hidden neurons. Left to right: ℓ_2 model recovery test error, MSE test loss, ℓ_∞ model recovery test error.



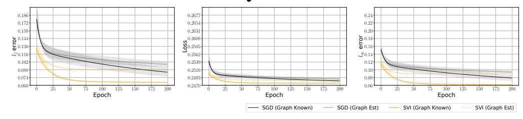
(d) Large graph, 4 hidden neurons. Left to right: ℓ_2 model recovery test error, MSE test loss, ℓ_∞ model recovery test error.



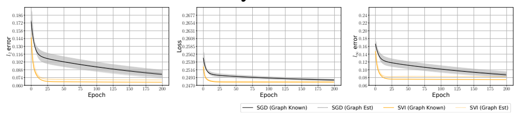
(e) Small graph, 8 hidden neurons. Left to right: ℓ_2 model recovery test error, MSE test loss, ℓ_∞ model recovery test error.



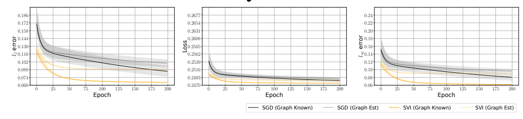
(f) Large graph, 8 hidden neurons. Left to right: ℓ_2 model recovery test error, MSE test loss, ℓ_∞ model recovery test error.



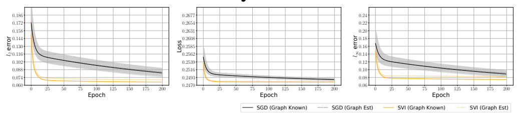
(g) Small graph, 16 hidden neurons. Left to right: ℓ_2 model recovery test error, MSE test loss, ℓ_∞ model recovery test error.



(h) Large graph, 16 hidden neurons. Left to right: ℓ_2 model recovery test error, MSE test loss, ℓ_∞ model recovery test error.



(i) Small graph, 32 hidden neurons. Left to right: ℓ_2 model recovery test error, MSE test loss, ℓ_∞ model recovery test error.

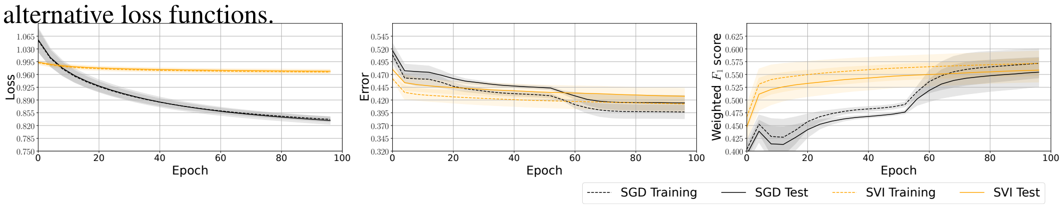


(j) Large graph, 32 hidden neurons. Left to right: ℓ_2 model recovery test error, MSE test loss, ℓ_∞ model recovery test error.

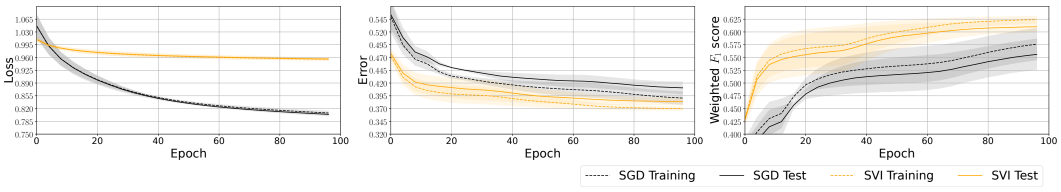
Figure 11: Two-layer GCN model recovery and prediction on small (left column) or large (left column) graphs. We compare SGD vs. `SVI` under different numbers of hidden neurons with $B = 100$. `SVI` consistently reaches smaller error with faster convergence. Metrics are defined in (12) and (16) respectively.

B.4 THREE-LAYER REAL TRAFFIC DATA

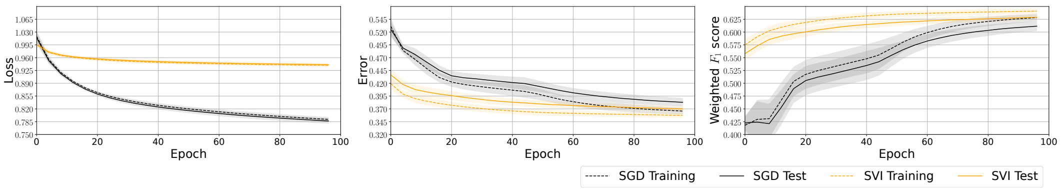
Traffic data. The raw bi-hourly traffic flow data are from the California Department of Transportation, where we collected data from 20 non-uniformly spaced traffic sensors in 2020`https://pems.dot.ca.gov/`. Data are available hourly, with $Y_{t,i} = 1$ (resp. 2) if the current traffic flow lies outside the upper (resp. lower) 90% quantile over the past four days of traffic flow of its nearest four neighbors based on sensor proximity. As before, we define feature X_t as the collection of past d days of observation and set $d = 4$, where the edges include the nearest five neighbors based on location. Data in the first nine months are training data (e.g., $N = 6138$) and the rest for testing ($N_1 = 2617$). We let $B = 600$ and $E = 100$ and use the cross-entropy loss to test the performance of `SVI` under alternative loss functions.



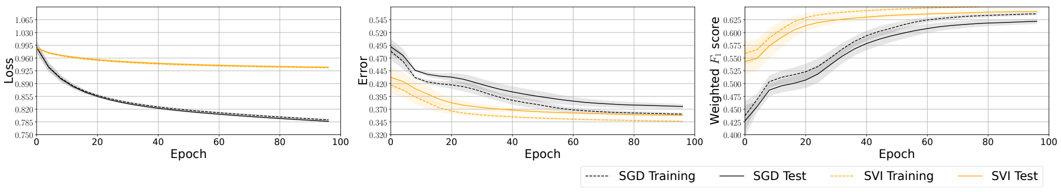
(a) 8 Hidden neurons for both hidden layers. Left to right: cross-entropy loss, classification error, and weighted F_1 score.



(b) 16 Hidden neurons for both hidden layers. Left to right: cross-entropy loss, classification error, and weighted F_1 score.



(c) 32 Hidden neurons for both hidden layers. Left to right: cross-entropy loss, classification error, and weighted F_1 score.



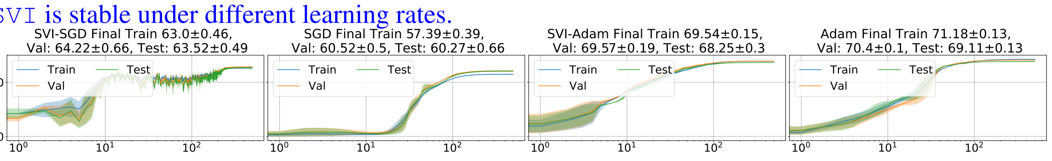
(d) 64 Hidden neurons for both hidden layers. Left to right: cross-entropy loss, classification error, and weighted F_1 score.

Figure 12: Traffic data multi-class anomaly detection under a three-layer GCN model. `SVI` shows faster convergence in terms of classification error and weighted F_1 scores. It reaches larger cross-entropy losses, which we think are benign (see Section 5.2 for justification).

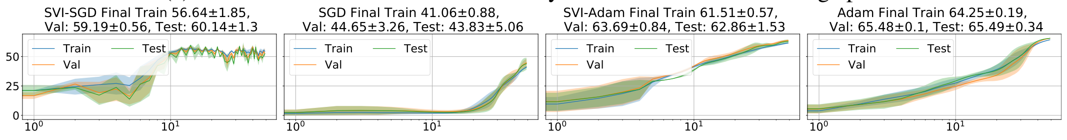
B.5 FOUR-LAYER OGB REAL-DATA

Data and model description. The graph nodes are papers to be classified into categories and edges denote citation among papers; it has ∼170 thousand nodes, 1.16 million edges, 128-dimensional node features, and 40 node classes. This graph is significantly larger than earlier examples. We train four-layer GCN models with hidden nodes kept at 512, which are both wider and deeper than earlier models.

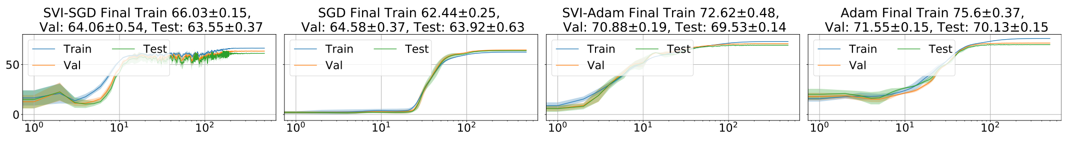
Additional results. We conduct further experiments when the number of hidden neurons in each hidden layer is reduced to 128 or 256. Figure 13 shows improved initial convergence by `SVI` or `SVI`-Adam over the competitors, a pattern consistently observed earlier, and competitive overall accuracies. In particular, SVI is consistent under different number of hidden neurons whereas SGD performs clearly worse under 128 hidden neurons. We also show in Table 6 that the performance of SVI is stable under different learning rates.



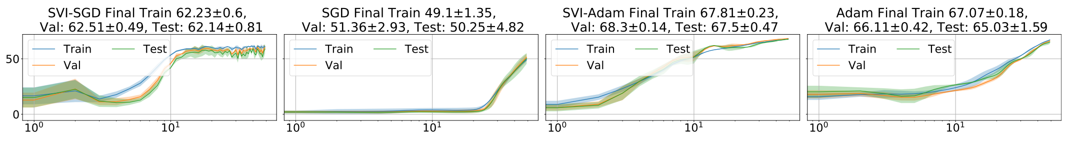
(a) 128 Hidden neurons for all hidden layers. Results over all training epochs



(b) 128 Hidden neurons for all hidden layers. Results over initial 50 training epochs



(c) 256 Hidden neurons for all hidden layers. Results over all training epochs



(d) 256 Hidden neurons for all hidden layers. Results over initial 50 training epochs

Figure 13: OGB large-scale real-data example under either 128 or 256 hidden neurons for all hidden layers. The setup is identical to Figure 7. It is clear that SVI or SVI-Adam still exhibits comparable final accuracy and improved convergence at initial training stages under such variants. In particular, SGD is sensitive to the number of hidden neurons, as its final accuracies under 128 hidden neurons are clearly poorer than those by SVI and very different from SGD performance under 256 hidden neurons. In contrast, SVI or SVI-Adam performs more consistently.

Final	SVI-SGD			SGD			SVI-Adam			Adam		
	Train	Valid	Test	Train	Valid	Test	Train	Valid	Test	Train	Valid	Test
0.005	75.85 (0.07)	71.73 (0.1)	70.41 (0.16)	75.01 (0.02)	71.3 (0.05)	69.91 (0.08)	77.31 (1.02)	71.53 (0.17)	70.65 (0.53)	78.28 (1.12)	71.26 (0.11)	70.5 (0.23)
0.01	77.17 (0.37)	71.87 (0.06)	70.58 (0.14)	76.83 (0.32)	71.74 (0.07)	70.33 (0.14)	76.04 (0.35)	71.21 (0.17)	70.69 (0.29)	76.04 (1.15)	71.35 (0.12)	70.23 (0.06)

Initial	SVI-SGD			SGD			SVI-Adam			Adam		
	Train	Valid	Test	Train	Valid	Test	Train	Valid	Test	Train	Valid	Test
0.005	63.76 (0.91)	63.48 (1.5)	63.91 (1.43)	62.05 (0.52)	62.7 (1.55)	62.45 (2.59)	66.26 (0.45)	65.64 (0.06)	65.58 (0.12)	67.58 (0.21)	66.67 (0.63)	66.2 (0.8)
0.01	64.76 (0.99)	64.06 (1.39)	64.22 (1.58)	63.46 (0.43)	63.2 (1.36)	63.64 (1.83)	67.41 (0.25)	67.49 (0.68)	67.17 (0.77)	67.56 (0.59)	67.28 (0.66)	67.12 (1.09)

Table 6: Classification accuracies on the large ogbn-arxiv dataset with larger learning rates by SVI and SGD. The set up is identical to Figure 7. The performance of SVI is stable under larger learning rate, with the same observed improvement over gradient-based method.