Ordered SGD: A New Stochastic Optimization Framework for Empirical Risk Minimization

Kenji Kawaguchi[*]
MIT

Haihao Lu[*]
Google Research

Abstract

We propose a new stochastic optimization framework for empirical risk minimization problems such as those that arise in machine learning. The traditional approaches, such as (mini-batch) stochastic gradient descent (SGD), utilize an unbiased gradient estimator of the empirical average loss. In contrast, we develop a computationally efficient method to construct a gradient estimator that is purposely biased toward those observations with higher current losses. On the theory side, we show that the proposed method minimizes a new ordered modification of the empirical average loss, and is guaranteed to converge at a sublinear rate to a global optimum for convex loss and to a critical point for weakly convex (non-convex) loss. Furthermore, we prove a new generalization bound for the proposed algorithm. On the empirical side, the numerical experiments show that our proposed method consistently improves the test errors compared with the standard mini-batch SGD in various models including SVM, logistic regression, and deep learning problems.

1 Introduction

Stochastic Gradient Descent (SGD), as the workhorse training algorithm for most machine learning applications including deep learning, has been extensively studied in recent years (e.g., see a recent review by Bottou et al. 2018). At every step, SGD draws one training sample uniformly at random from the training dataset, and then uses the (sub-)gradient of the

loss over the selected sample to update the model parameters. The most popular version of SGD in practice is perhaps the mini-batch SGD (Bottou et al., 2018; Dean et al., 2012), which is widely implemented in the state-of-the-art deep learning frameworks, such as TensorFlow (Abadi et al., 2016), PyTorch (Paszke et al., 2017) and CNTK (Seide and Agarwal, 2016). Instead of choosing one sample per iteration, mini-batch SGD randomly selects a mini-batch of the samples, and uses the (sub-)gradient of the average loss over the selected samples to update the model parameters.

Both SGD and mini-batch SGD utilize uniform sampling during the entire learning process, so that the stochastic gradient is always an unbiased gradient estimator of the empirical average loss over all samples. On the other hand, it appears to practitioners that not all samples are equally important, and indeed most of them could be ignored after a few epochs of training without affecting the final model (Katharopoulos and Fleuret, 2018). For example, intuitively, the samples near the final decision boundary should be more important to build the model than those far away from the boundary for classification problems. In particular, as we will illustrate later in Figure 1, there are cases when those far-away samples may corrupt the model by using average loss. In order to further explore such structures, we propose an efficient sampling scheme on top of the mini-batch SGD. We call the resulting algorithm *ordered SGD*, which is used to learn a different type of models with the goal to improve the testing performance.

The above motivation of ordered SGD is related to that of importance sampling SGD, which has been extensively studied recently in order to improve the convergence speed of SGD (Needell et al., 2014; Zhao and Zhang, 2015; Alain et al., 2015; Loshchilov and Hutter, 2015; Gopal, 2016; Katharopoulos and Fleuret, 2018). However, our goals, algorithms and theoretical results are fundamentally different from those in the previous studies on importance sampling SGD. Indeed, all aforementioned studies are aimed to accelerate the

---

[*]equal contribution

minimization process for the empirical average loss, whereas our proposed method turns out to minimize a new objective function by purposely constructing a biased gradient.

Our main contributions can be summarized as follows: i) we propose a computationally efficient and easily implementable algorithm, ordered SGD, with principled motivations (Section 3), ii) we show that ordered SGD minimizes an ordered empirical risk with sublinear rate for convex and weakly convex (non-convex) loss functions (Section 4), iii) we prove a generalization bound for ordered SGD (Section 5), and iv) our numerical experiments show ordered SGD consistently improved mini-batch SGD in test errors (Section 6).

2 Empirical Risk Minimization

Empirical risk minimization is one of the main tools to build a model in machine learning. Let $\mathcal{D} = ((x_i, y_i))_{i=1}^n$ be a training dataset of n samples where $x_i \in \mathcal{X} \subseteq \mathbb{R}^{d_x}$ is the input vector and $y_i \in \mathcal{Y} \subseteq \mathbb{R}^{d_y}$ is the target output vector for the i-th sample. The goal of empirical risk minimization is to find a prediction function $f(\cdot\,; \theta) : \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}$, by minimizing

$$L(\theta) := \frac{1}{n} \sum_{i=1}^n L_i(\theta) + R(\theta), \qquad (1)$$

where $\theta \in \mathbb{R}^{d_\theta}$ is the parameter vector of the prediction model, $L_i(\theta) := \ell(f(x_i; \theta), y_i)$ with the function $\ell : \mathbb{R}^{d_y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$ is the loss of the i-th sample, and $R(\theta) \geq 0$ is a regularizer. For example, in logistic regression, $f(x; \theta) = \theta^T x$ is a linear function of the input vector x, and $\ell(a, y) = \log(1 + \exp(-ya))$ is the logistic loss function with $y \in \{-1, 1\}$. For a neural network, $f(x; \theta)$ represents the pre-activation output of the last layer.

3 Algorithm

In this section, we introduce ordered SGD and provide an intuitive explanation of the advantage of ordered SGD by looking at 2-dimension toy examples with linear classifiers and small artificial neural networks (ANNs). Let us first introduce a new notation q-argmax as an extension to the standard notation argmax:

Definition 1. Given a set of n real numbers $(a_1, a_2, \ldots, a_n)$, an index subset $S \subseteq \{1, 2, \ldots, n\}$, and a positive integer number $q \leq |S|$, we define q-argmax$_{j \in S}\, a_j$ such that $Q \in$ q-argmax$_{j \in S}\, a_j$ is a set of q indexes of the q largest values of $(a_j)_{j \in S}$; i.e., q-argmax$_{j \in S}\, a_j = \operatorname{argmax}_{Q \subseteq S, |Q|=q} \sum_{i \in Q} a_i$.

Algorithm 1 Ordered Stochastic Gradient Descent (ordered SGD)

1: **Inputs:** an initial vector θ^0 and a learning rate sequence $(\eta_k)_k$
2: **for** $t = 1, 2, \ldots$ **do**
3:     Randomly choose a mini-batch of samples: $S \subseteq \{1, 2, \ldots, n\}$ such that $|S| = s$.
4:     Find a set Q of top-q samples in S in term of loss values: $Q \in$ q-argmax$_{i \in S} L_i(\theta^t)$.
5:     Compute a subgradient $\tilde{g}^t$ of the top-q samples $L_Q(\theta^t)$: $\tilde{g}^t \in \partial L_Q(\theta^t)$ where $L_Q(\theta^t) = \frac{1}{q} \sum_{i \in Q} L_i(\theta^t) + R(\theta^t)$ and ∂L_Q is the set of subgradient[1] of function L_Q.
6:     Update parameters θ: $\theta^{t+1} = \theta^t - \eta_t \tilde{g}^t$

Algorithm 1 describes the pseudocode of our proposed algorithm, ordered SGD. The procedures of ordered SGD follow those of mini-batch SGD except the following modification: after drawing a mini-batch of size s, ordered SGD updates the parameter vector θ based on the (sub-)gradient of the average loss over the top-q samples in the mini-batch in terms of individual loss values (lines 4 and 5 of Algorithm 1). This modification is used to purposely build and utilize a biased gradient estimator with more weights on the samples having larger losses. As it can be seen in Algorithm 1, ordered SGD is easily implementable, requiring to change only a single line or few lines on top of a mini-batch SGD implementation.

Figure 1 illustrates the motivation of ordered SGD by looking at two-dimensional toy problems of binary classification. To avoid an extra freedom due to the hyper-parameter q, we employed a single fixed procedure to set the hyper-parameter q in the experiments for Figure 1 and other experiments in Section 6, which is further explained in Section 6. The details of the experimental settings for Figure 1 are presented in Section 6 and in Appendix C.

It can be seen from Figure 1 that ordered SGD adapts better to imbalanced data distributions compared with mini-batch SGD. It can better capture the information of the smaller sub-clusters that contribute less to the empirical average loss $L(\theta)$: e.g., the small sub-clusters in the middle of Figures 1a and 1b, as well as the small inner ring structure in Figures 1c and 1d (the two inner rings contain only 40 data points while the two outer rings contain 960 data points). The smaller sub-clusters are informative for training a classifier when they are not outliers or by-products of noise. A sub-

---

[1]The sub-gradient for (non-convex) ρ-weakly convex function L_Q at θ^t is defined as $\{g \mid L_Q(\theta) \geq L_Q(\theta^t) + \langle g, \theta - \theta^t \rangle - \frac{\rho}{2}\|\theta - \theta^t\|^2, \forall\theta\}$ (Rockafellar and Wets, 2009).

Kenji Kawaguchi[*], Haihao Lu[*]

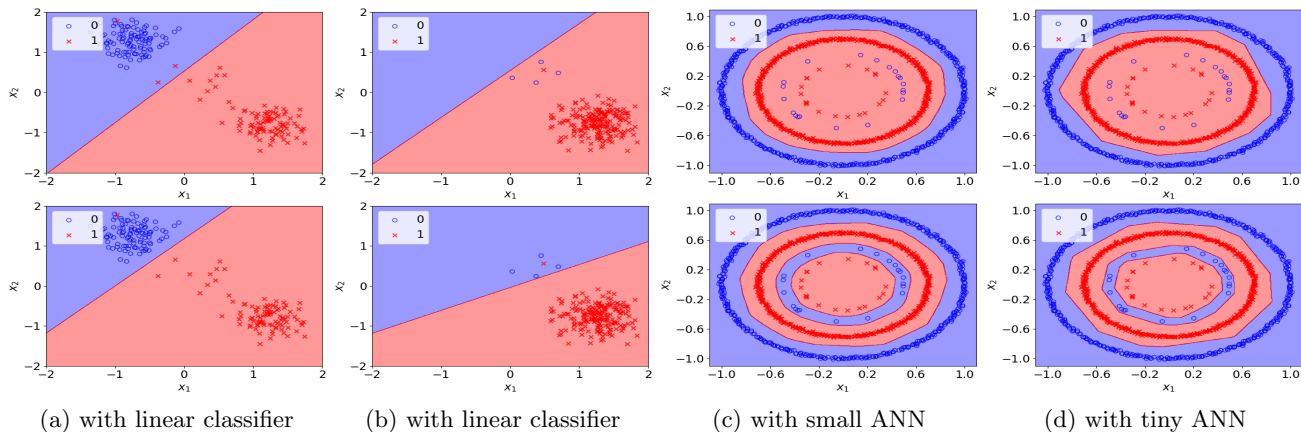(a) with linear classifier   (b) with linear classifier   (c) with small ANN   (d) with tiny ANN

Figure 1: Decision boundaries of mini-batch SGD predictors (**top** row) and ordered SGD predictors (**bottom** row) with 2D synthetic datasets for binary classification. In these examples, ordered SGD predictors correctly classify more data points than mini-batch SGD predictors, because a ordered SGD predictor can focus more on a smaller yet informative subset of data points, instead of focusing on the average loss dominated by a larger subset of data points.

cluster of data points would be less likely to be an outlier as the size of the sub-cluster increases. The value of q in ordered SGD can control the size of sub-clusters that a classifier should be sensitive to. With smaller q, the output model becomes more sensitive to smaller sub-clusters. In an extreme case with $q = 1$ and $n = s$, ordered SGD minimizes the maximal loss (Shalev-Shwartz and Wexler, 2016) that is highly sensitive to every smallest sub-cluster of each single data point.

4   Optimization Theory

In this section, we answer the following three questions: (1) what objective function does ordered SGD solve as an optimization method, (2) what is the convergence rate of ordered SGD for minimizing the new objective function, and (3) what is the asymptotic structure of the new objective function.

Similarly to the notation of order statistics, we first introduce the notation of ordered indexes: given a model parameter θ, let $L_{(1)}(\theta) \geq L_{(2)}(\theta) \geq \cdots \geq L_{(n)}(\theta)$ be the decreasing values of the individual losses $L_1(\theta), \ldots, L_n(\theta)$, where $(j) \in \{1, \ldots, n\}$ (for all $j \in \{1, \ldots, n\}$). That is, $\{(1), \ldots, (n)\}$ as a perturbation of $\{1, \ldots, n\}$ defines the order of sample indexes by loss values. Throughout this paper, whenever we encounter ties on the values, we employ a tie-breaking rule in order to ensure the uniqueness of such an order.[2] Theorem 1 shows that ordered SGD is a stochastic first-order method for minimizing the new ordered

---

[2]In the case of ties, the order is defined by the order of the original indexes $(1, 2, \ldots, n)$ of $L_1(\theta), \ldots, L_n(\theta)$; i.e., if $L_{i_1}(\theta) = L_{i_2}(\theta)$ and $i_1 < i_2$, then i_1 appears before i_2 in the sequence $((1), (2), \ldots, (n))$.

empirical loss $L_q(\theta)$.

Theorem 1. *Consider the following objective function:*

$$L_q(\theta) := \frac{1}{q} \sum_{j=1}^{n} \gamma_j L_{(j)}(\theta) + R(\theta), \qquad (2)$$

where the parameter γ_j depends on the tuple (n, s, q), and is defined by

$$\gamma_j := \frac{\sum_{l=0}^{q-1} \binom{j-1}{l}\binom{n-j}{s-l-1}}{\binom{n}{s}}. \qquad (3)$$

Then, ordered SGD is a stochastic first-order method for minimizing $L_q(\theta)$ in the sense that $\tilde{g}^t$ used in ordered SGD is an unbiased estimator of a (sub-)gradient of $L_q(\theta)$.

Although the order of individual losses change with different θ, L_q is a well-defined function. For any given θ, the order of individual losses is fixed and $L_q(\theta)$ has a unique value, which means $L_q(\theta)$ is a function of θ.

All proofs in this paper are deferred to Appendix A. As we can see from Theorem 1, the objective function minimized by ordered SGD (i.e., $L_q(\theta)$) depends on the hyper-parameters of the algorithm through the values of γ_j. Therefore, it is of practical interest to obtain deeper understandings on how the hyper-parameters (n, s, q) affects the objective function $L_q(\theta)$ through γ_j. The next proposition presents the asymptotic value of γ_j (when $n \to \infty$), which shows that a rescaled γ_j converges to the cumulative distribution function of a Beta distribution:

Proposition 1. *Denote $z = \frac{j}{n}$ and $\gamma(z) := \sum_{l=0}^{q-1} z^l (1-z)^{s-l-1} \frac{s!}{l!(s-l-1)!}$. Then, it holds that*

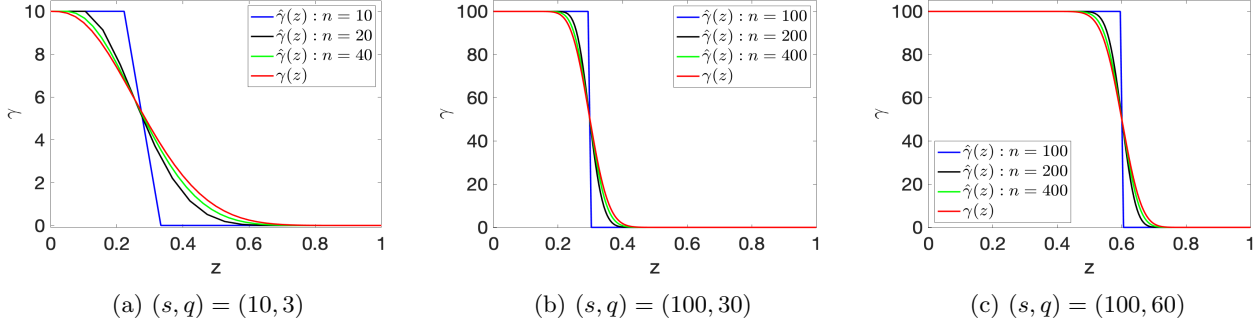$$\lim_{j,n \to \infty, j/n=z} \gamma_j = \frac{1}{n}\gamma(z).$$

Figure 2: $\hat{\gamma}(z)$ and $\gamma(z)$ for different (n, s, q) where $\hat{\gamma}$ is a rescaled version of γ_j: $\hat{\gamma}(j/n) = n\gamma_j$.

Moreover, it holds that $1 - \frac{1}{s}\gamma(z)$ is the cumulative distribution function of $Beta(z; q, s - q)$.

To better illustrate the structure of γ_j in the non-asymptotic regime, Figure 2 plots $\hat{\gamma}(z)$ and $\gamma(z)$ for different values of (n, s, q) where $\hat{\gamma}(z)$ is a rescaled version of γ_j defined by $\hat{\gamma}(j/n) = n\gamma_j$ (and the value of $\hat{\gamma}(\cdot)$ between j/n and $(j + 1)/n$ is defined by linear interpolation for better visualization). As we can see from Figure 2, $\hat{\gamma}(z)$ monotonically decays. In each subfigure, with fixed s, q, the cliff gets smoother and $\hat{\gamma}(z)$ converges to $\gamma(z)$ as n increases. Comparing Figures 2a and 2b, we can see that as s, q and n all increase proportionally, the cliff gets steeper. Comparing Figures 2b and 2c, we can see that with fixed n and q, the cliff shifts to the right as q increases.

As a direct extension of Theorem 1, we can now obtain the computational guarantees of ordered SGD for minimizing $L_q(\theta)$ by taking advantage of the classic convergence results of SGD:

Theorem 2. *Let* $(\theta^t)_{t=0}^T$ *be a sequence generated by ordered SGD (Algorithm 1). Suppose that* $L_i(\cdot)$ *is* G_1-*Lipschitz continuous for* $i = 1, \ldots, n$, *and* $R(\cdot)$ *is* G_2-*Lipschitz continuous. Suppose that there exists a finite* $\theta^* \in \operatorname{argmin}_\theta L_q(\theta)$ *and* $L_q(\theta^*)$ *is finite. Then, the following two statements hold:*

(1) *(Convex setting). If* $L_i(\cdot)$ *and* $R(\cdot)$ *are both convex, for any step-size* η_t, *it holds that*

$$\min_{0 \leq t \leq n} \mathbb{E}[L_q(\theta^t) - L_q(\theta^*)]$$
$$\leq \frac{2(G_1^2 + G_2^2)\sum_{t=0}^T \eta_t^2 + \|\theta^* - \theta^0\|^2}{2\sum_{t=0}^T \eta_t} .$$

(2) *(Weakly convex setting) Suppose that* $L_i(\cdot)$ *is* ρ-*weakly convex (i.e.,* $L_i(\theta) + \frac{\rho}{2}\|\theta\|^2$ *is convex) and* $R(\cdot)$ *is convex. Recall the definition of Moreau envelope:* $L_q^\lambda(\theta) := \min_\beta \{L_q(\beta) + \frac{1}{2\lambda}\|\beta - \theta\|^2\}$. *Denote* $\bar{\theta}^T$ *as a random variable taking value in* $\{\theta^0, \theta^1, \ldots, \theta^T\}$ *according to the probability distri-*

bution $\mathbb{P}(\bar{\theta}^T = \theta^t) = \frac{\eta_t}{\sum_{t=0}^T \eta_t}$. *Then for any constant* $\hat{\rho} > \rho$, *it holds that*

$$\mathbb{E}[\|\nabla L_q^{1/\hat{\rho}}(\bar{\theta}^T)\|^2]$$
$$\leq \frac{\hat{\rho}}{\hat{\rho} - \rho} \frac{\left(L_q^{1/\hat{\rho}}(\theta^0) - L_q(\theta^*)\right) + \hat{\rho}(G_1^2 + G_2^2)\sum_{t=0}^T \eta_t^2}{\sum_{t=0}^T \eta_t}.$$

Theorem 2 shows that in particular, if we choose $\eta_t \sim O(1/\sqrt{t})$, the optimality gap $\min_t L_q(\theta^t) - L_q(\theta^*)$ and $\mathbb{E}[\|\nabla L_q^{1/\hat{\rho}}(\bar{\theta}_T)\|^2]$ decay at the rate of $\tilde{O}(1/\sqrt{t})$ (note that $\lim_{T \to \infty} \frac{\sum_{t=0}^T \eta_t^2}{\sum_{t=0}^T \eta_t} = 0$ with $\eta_t \sim O(1/\sqrt{t})$).

The Lipschitz continuity assumption in Theorem 2 is a standard assumption for the analysis of stochastic optimization algorithms. This assumption is generally satisfied with logistic loss, hinge loss and Huber loss without any constraints on θ^t, and with square loss when one can presume that θ^t stays in a compact space (which is typically the case being interested in practice). For the weakly convex setting, $\mathbb{E}\|\nabla \varphi^{1/2\rho}(\theta^k)\|^2$ (appeared in Theorem 2 (2)) is a natural measure of the near-stationarity for a non-differentiable weakly convex function $\varphi : \theta \mapsto \varphi(\theta)$ (Davis and Drusvyatskiy, 2018). The weak convexity (also known as negative strong convexity or almost convexity) is a standard assumption for analyzing non-convex optimization problem in optimization literature (Davis and Drusvyatskiy, 2018; Allen-Zhu, 2017). With a standard loss criterion such as logistic loss, the individual objective $L_i(\cdot)$ with a neural network using sigmoid or tanh activation functions is weakly convex (neural network with ReLU activation function is not weakly convex and falls out of our setting).

5   Generalization Bound

This section presents the generalization theory for ordered SGD. To make the dependence on a training dataset $\mathcal{D}$ explicit, we define $L(\theta; \mathcal{D}) := \frac{1}{n}\sum_{i=1}^n L_i(\theta; \mathcal{D})$ and $L_q(\theta; \mathcal{D}) := \frac{1}{q}\sum_{j=1}^m \gamma_j L_{(j)}(\theta; \mathcal{D})$ by rewriting $L_i(\theta; \mathcal{D}) = L_i(\theta)$ and $L_{(j)}(\theta; \mathcal{D}) =$

Kenji Kawaguchi[*], Haihao Lu[*]

$L_{(j)}(\theta)$, where $((j))_{j=1}^n$ defines the order of sample indexes by the loss value, as stated in Section 4. Denote $r_i(\theta; \mathcal{D}) = \sum_{j=1}^n \mathbb{1}\{i = (j)\}\gamma_j$ where (j) depends on $(\theta, \mathcal{D})$. Given an arbitrary set $\Theta \subseteq \mathbb{R}^{d_\theta}$, we define $\mathfrak{R}_n(\Theta)$ as the (standard) Rademacher complexity of the set $\{(x, y) \mapsto \ell(f(x; \theta), y) : \theta \in \Theta\}$:

$$\mathfrak{R}_n(\Theta) = \mathbb{E}_{\bar{\mathcal{D}}, \xi}\left[\sup_{\theta \in \Theta} \frac{1}{n}\sum_{i=1}^n \xi_i \ell(f(\bar{x}_i; \theta), \bar{y}_i)\right],$$

where $\overline{\mathcal{D}} = ((\bar{x}_i, \bar{y}_i))_{i=1}^n$, and $\xi_1, \ldots, \xi_n$ are independent uniform random variables taking values in $\{-1, 1\}$ (i.e., Rademacher variables). Given a tuple $(\ell, f, \Theta, \mathcal{X}, \mathcal{Y})$, define M as the least upper bound on the difference of individual loss values: $|\ell(f(x; \theta), y) - \ell(f(x'; \theta), y')| \le M$ for all $\theta \in \Theta$ and all $(x, y), (x', y') \in \mathcal{X} \times \mathcal{Y}$. For example, $M = 1$ if ℓ is the 0-1 loss function. Theorem 3 presents a generalization bound for ordered SGD:

Theorem 3. *Let Θ be a fixed subset of $\mathbb{R}^{d_\theta}$. Then, for any $\delta > 0$, with probability at least $1 - \delta$ over an iid draw of n examples $\mathcal{D} = ((x_i, y_i))_{i=1}^n$, the following holds for all $\theta \in \Theta$:*

$$\mathbb{E}_{(x,y)}[\ell(f(x; \theta), y)] \tag{4}$$

$$\le L_q(\theta; \mathcal{D}) + 2\mathfrak{R}_n(\Theta) + \frac{Ms}{q}\sqrt{\frac{\ln(1/\delta)}{2n}} - \mathcal{Q}_n(\Theta; s, q),$$

where $\mathcal{Q}_n(\Theta; s, q) := \mathbb{E}_{\bar{\mathcal{D}}}[\inf_{\theta \in \Theta} \sum_{i=1}^n (\frac{r_i(\theta; \bar{\mathcal{D}})}{q} - \frac{1}{n})\ell(f(\bar{x}_i; \theta), \bar{y}_i)] \ge 0$.

The expected error $\mathbb{E}_{(x,y)}[\ell(f(x; \theta), y)]$ in the left-hand side of Equation (4) is a standard objective for generalization, whereas the right-hand side is an upper bound with the dependence on the algorithm parameters q and s. Let us first look at the asymptotic case when $n \to \infty$. Let Θ be constrained such that $\mathfrak{R}_n(\Theta) \to 0$ as $n \to \infty$, which has been shown to be satisfied for various models and sets Θ (Bartlett and Mendelson, 2002; Mohri et al., 2012; Bartlett et al., 2017; Kawaguchi et al., 2017). With s/q being bounded, the third term in the right-hand side of Equation (4) disappear as $n \to \infty$. Thus, it holds with high probability that $\mathbb{E}_{(x,y)}[\ell(f(x; \theta), y)] \le L_q(\theta; \mathcal{D}) - \mathcal{Q}_n(\Theta; s, q) \le L_q(\theta; \mathcal{D})$, where $L_q(\theta; \mathcal{D})$ is minimized by ordered SGD as shown in Theorem 1 and Theorem 2. From this viewpoint, ordered SGD minimizes the expected error for generalization when $n \to \infty$.

A special case of Theorem 3 recovers the standard generalization bound of the empirical average loss (e.g., Mohri et al., 2012), That is, if $q = s$, ordered SGD becomes the standard mini-batch SGD and Equation

(4) becomes

$$\mathbb{E}_{(x,y)}[\ell(f(x; \theta), y)] \le L(\theta; \mathcal{D}) + 2\mathfrak{R}_n(\Theta) + M\sqrt{\frac{\ln\frac{1}{\delta}}{2n}}, \tag{5}$$

which is the standard generalization bound (e.g., Mohri et al., 2012). This is because if $q = s$, then $\frac{r_i(\theta; \bar{\mathcal{D}})}{q} = \frac{1}{n}$ and hence $\mathcal{Q}_n(\Theta; s, q) = 0$.

For the purpose of a simple comparison of ordered SGD and (mini-batch) SGD, consider the case where we fix a single subset $\Theta \subseteq \mathbb{R}^{d_\theta}$. Let $\hat{\theta}_q$ and $\hat{\theta}_s$ be the parameter vectors obtained by ordered SGD and (mini-batch) SGD respectively as the results of training. Then, when $n \to \infty$, with s/q being bounded, the upper bound on the expected error for ordered SGD (the right hand-side of Equation 4) is (strictly) less than that for (mini-batch) SGD (the right hand-side of Equation 5) if $\mathcal{Q}_n(\Theta; s, q) + L(\hat{\theta}_s; \mathcal{D}) - L_q(\hat{\theta}_q; \mathcal{D}) > 0$ or if $L(\hat{\theta}_s; \mathcal{D}) - L_q(\hat{\theta}_q; \mathcal{D}) > 0$.

For a given model f, whether Theorem 3 provides a non-vacuous bound depends on the choice of Θ. In Appendix B, we discuss this effect as well as a standard way to derive various data-dependent bounds from Theorem 3.

6 Experiments

In this section, we empirically evaluate ordered SGD with various datasets, models and settings. To avoid an extra freedom due to the hyper-parameter q, we introduce a single fixed setup of the adaptive values of q as the default setting: $q = s$ at the beginning of training, $q = \lfloor s/2 \rfloor$ once train_acc $\ge 80\%$, $q = \lfloor s/4 \rfloor$ once train_acc $\ge 90\%$, $q = \lfloor s/8 \rfloor$ once train_acc $\ge 95\%$, and $q = \lfloor s/16 \rfloor$ once train_acc $\ge 99.5\%$, where train_acc represents training accuracy. The value of q was automatically updated at the end of each epoch based on this simple rule. This rule was derived based on the intuition that in the early stage of training, all samples are informative to build a rough model, while the samples around the boundary (with larger losses) are more helpful to build the final classifier in later stage. In the figures and tables of this section, we refer to ordered SGD with this rule as 'OSGD', and ordered SGD with a fixed value $q = \bar{q}$ as 'OSGD: $q = \bar{q}$'.

Experiment with fixed hyper-parameters. For this experiment, we fixed all hyper-parameters a priori across all different datasets and models by using a standard hyper-parameter setting of mini-batch SGD, instead of aiming for state-of-the-art test errors for each dataset with a possible issue of over-fitting to test and validation datasets (Dwork et al., 2015; Rao et al., 2008). We fixed the mini-batch size s to be 64,

Table 1: Test errors (%) of mini-batch SGD and ordered SGD (OSGD). The last column labeled "Improve" shows relative improvements (%) from mini-batch SGD to ordered SGD. In the other columns, the numbers indicate the mean test errors (and standard deviations in parentheses) over ten random trials. The first column shows 'No' for no data augmentation, and 'Yes' for data augmentation.

Data Aug	Datasets	Model	mini-batch SGD	OSGD	Improve
No	Semeion	Logistic model	10.76 (0.35)	9.31 (0.42)	13.48
No	MNIST	Logistic model	7.70 (0.06)	7.35 (0.04)	4.55
No	Semeion	SVM	11.05 (0.72)	10.25 (0.51)	7.18
No	MNIST	SVM	8.04 (0.05)	7.66 (0.07)	4.60
No	Semeion	LeNet	8.06 (0.61)	6.09 (0.55)	24.48
No	MNIST	LeNet	0.65 (0.04)	0.57 (0.06)	11.56
No	KMNIST	LeNet	3.74 (0.08)	3.09 (0.14)	17.49
No	Fashion-MNIST	LeNet	8.07 (0.16)	8.03 (0.26)	0.57
No	CIFAR-10	PreActResNet18	13.75 (0.22)	12.87 (0.32)	6.41
No	CIFAR-100	PreActResNet18	41.80 (0.40)	41.32 (0.43)	1.17
No	SVHN	PreActResNet18	4.66 (0.10)	4.39 (0.11)	5.95
Yes	Semeion	LeNet	7.47 (1.03)	5.06 (0.69)	32.28
Yes	MNIST	LeNet	0.43 (0.03)	0.39 (0.03)	9.84
Yes	KMNIST	LeNet	2.59 (0.09)	2.01 (0.13)	22.33
Yes	Fashion-MNIST	LeNet	7.45 (0.07)	6.49 (0.19)	12.93
Yes	CIFAR-10	PreActResNet18	8.08 (0.17)	7.04 (0.12)	12.81
Yes	CIFAR-100	PreActResNet18	29.95 (0.31)	28.31 (0.41)	5.49
Yes	SVHN	PreActResNet18	4.45 (0.07)	4.00 (0.08)	10.08

the weight decay rate to be 10^{-4}, the initial learning rate to be 0.01, and the momentum coefficient to be 0.9. See Appendix C for more details of the experimental settings. The code to reproduce all the results is publicly available at: [the link is hidden for anonymous submission].

Table 1 compares the testing performance of ordered SGD and mini-batch SGD for different models and datasets. Table 1 consistently shows that ordered SGD improved mini-batch SGD in test errors. The table reports the mean and the standard deviation of test errors (i.e., 100 × the average of 0-1 losses on test dataset) over 10 random experiments with different random seeds. The table also summarises the relative improvements of ordered SGD over mini-batch SGD, which is defined as [100× ((mean test error of mini-batch SGD) - (mean test error of ordered SGD)) / (mean test error of mini-batch SGD)]. Logistic model refers to linear multinomial logistic regression model, SVM refers to linear multiclass support vector machine, LeNet refers to a standard variant of LeNet (LeCun et al., 1998) with ReLU activations, and PreActResNet18 refers to pre-activation ResNet with 18 layers (He et al., 2016).

Figure 3 shows the test error and the average training loss of mini-batch SGD and ordered SGD versus the number of epoch. As shown in the figure, ordered SGD with the fixed q value also outperformed mini-batch SGD in general. In the figures, the reported training losses refer to the standard empirical average loss $\frac{1}{n}\sum_{i=1}^{n} L_i(\theta)$ measured at the end of each epoch. When compared to mini-batch SGD, ordered SGD had lower test errors *while having higher training losses* in Figures 3a, 3d and 3g, because ordered SGD optimizes over the ordered empirical loss instead. This is consistent with our motivation and theory of ordered SGD in Sections 3, 4 and 5. The qualitatively similar behaviors were also observed with all of the 18 various problems as shown in Appendix C.

Moreover, ordered SGD is a computationally efficient algorithm. Table 2 shows the wall-clock time in illustrative four experiments, whereas Table 4 in Appendix C summarizes the wall-clock time in all experiments. The wall-clock time of ordered SGD measures the time spent by all computations of ordered SGD, including the extra computation of finding top-q samples in a mini-batch (line 4 of Algorithm 1). The extra computation is generally negligible and can be completed in $O(s \log q)$ or $O(s)$ by using a sorting/selection algorithm. The ordered SGD algorithm can be faster than mini-batch SGD because ordered SGD only computes the (sub-)gradient $\tilde{g}^t$ of the top-q samples (in
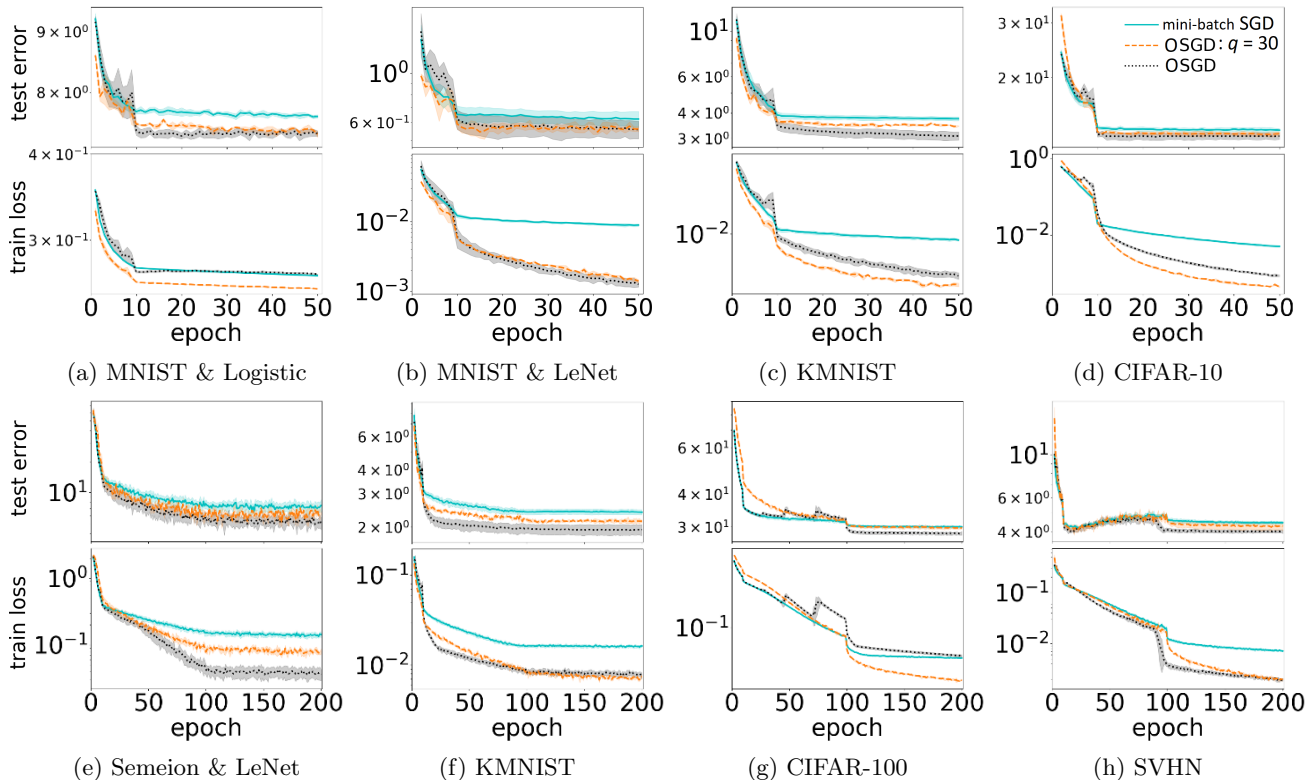
Kenji Kawaguchi[*], Haihao Lu[*]

Figure 3: Test error and training loss (in log scales) versus the number of epoch. These are without data augmentation in subfigures (a)-(d), and with data augmentation in subfigures (e)-(h). The lines indicate the mean values over 10 random trials, and the shaded regions represent intervals of the sample standard deviations.

Table 2: Average wall-clock time (seconds) per epoch with data augmentation. PreActResNet18 was used for CIFAR-10, CIFAR-100, and SVHN, while LeNet was used for MNIST and KMNIST.

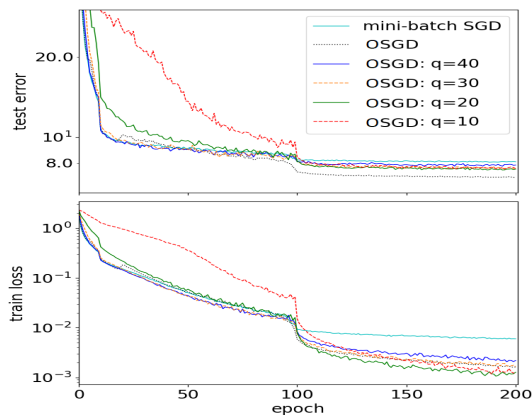| Datasets	mini-batch SGD	OSGD
MNIST	14.44 (0.54)	14.77 (0.41)
KMNIST	12.17 (0.33)	11.42 (0.29)
CIFAR-10	48.18 (0.58)	46.40 (0.97)
CIFAR-100	47.37 (0.84)	44.74 (0.91)
SVHN	72.29 (1.23)	67.95 (1.54)

Figure 4: Effect of different q values with CIFAR-10.

line 5 of Algorithm 1). As shown in Tables 2 and 4, ordered SGD was faster than mini-batch SGD for all larger models with PreActResNet18. This is because the computational reduction of the back-propagation in ordered SGD can dominate the small extra cost of finding top-q samples in larger problems.

Experiment with different q values. Figure 4 shows the effect of different fixed q values for CIFAR-10 with PreActResNet18. Ordered SGD improved the

test errors of mini-batch SGD with different fixed q values. We also report the same observation with different datasets and models in Appendix C.

Experiment with different learning rates and mini-batch sizes. Figures 5 and 6 in Appendix C consistently show the improvement of ordered SGD over mini-batch SGD with different different learning rates and mini-batch sizes.

Table 3: Test errors (%) by using the best learning rate of mini-batch SGD with various data augmentation methods for CIFAR-10.

Data Aug	mini-batch SGD	OSGD	Improve
Standard	6.94	6.46	6.92
RE	3.24	3.06	5.56
Mixup	3.31	3.05	7.85

Experiment with the best learning rate, mixup, and random erasing. Table 3 summarises the experimental results with the data augmentation methods of random erasing (RE) (Zhong et al., 2017) and mixup (Zhang et al., 2017; Verma et al., 2019) by using CIFAR-10 dataset. For this experiment, we purposefully adopted the setting that favors mini-batch SGD. That is, for both mini-batch SGD and ordered SGD, we used hyper-parameters tuned for mini-batch SGD. For RE and mixup data, we used the same tuned hyper-parameter settings (including learning rates) and the codes as those in the previous studies that used mini-batch SGD (Zhong et al., 2017; Verma et al., 2019) (with WRN-28-10 for RE and with PreActRes-Net18 for mixup). For standard data augmentation, we first searched the best learning rate of mini-batch SGD based on the test error (purposefully overfitting to the test dataset for mini-batch SGD) by using the grid search with learning rates of 1.0, 0.5, 0.1, 0.05. 0.01, 0.005, 0.001, 0.0005, 0.0001. Then, we used the best learning rate of mini-batch SGD for ordered SGD (instead of using the best learning rate of ordered SGD for ordered SGD). As shown in Table 3, ordered SGD with hyper-parameters tuned for mini-batch SGD still outperformed fine-tuned mini-batch SGD with the different data augmentation methods.

7   Related work and extension

Although there is no direct predecessor of our work, the following fields are related to this paper.

Other mini-batch stochastic methods. The proposed sampling strategy and our theoretical analyses are generic and can be extended to other (mini-batch) stochastic methods, including Adam (Kingma and Ba, 2014), stochastic mirror descent (Beck and Teboulle, 2003; Nedic and Lee, 2014; Lu, 2017; Lu et al., 2018; Zhang and He, 2018), and proximal stochastic subgradient methods (Davis and Drusvyatskiy, 2018). Thus, our results open up the research direction for further studying the proposed stochastic optimization framework with different base algorithms such as Adam and AdaGrad. To illustrate it, we presented ordered Adam and reported the numerical results in Appendix C.

Importance Sampling SGD. Stochastic gradient descent with importance sampling has been an ac-

tive research area for the past several years (Needell et al., 2014; Zhao and Zhang, 2015; Alain et al., 2015; Loshchilov and Hutter, 2015; Gopal, 2016; Katharopoulos and Fleuret, 2018). In the convex setting, (Zhao and Zhang, 2015; Needell et al., 2014) show that the optimal sampling distribution for minimizing $L(\theta)$ is proportional to the per-sample gradient norm. However, maintaining the norm of gradient for individual samples can be computationally expensive when the dataset size n or the parameter vector size d_θ is large in particular for many applications of deep learning. These importance sampling methods are inherently different from ordered SGD in that importance sampling is used to reduce the number of iterations for minimizing $L(\theta)$, whereas ordered SGD is designed to learn a different type of models by minimizing the new objective function $L_q(\theta)$.

Average Top-k Loss. The average top-k loss is introduced by Fan et al. (2017) as an alternative to the empirical average loss $L(\theta)$. The ordered loss function $L_q(\theta)$ differs from the average top-k loss as shown in Section 4. Furthermore, our proposed framework is fundamentally different from the average top-k loss. First, the algorithms are different – the stochastic method proposed in Fan et al. (2017) utilizes duality of the function and is unusable for deep neural networks (and other non-convex problems), while our proposed method is a modification of mini-batch SGD that is usable for deep neural networks (and other non-convex problems) and scales well for large problems. Second, the optimization results are different, and in particular, the objective functions are different and we have convergence analysis for weakly convex (non-convex) functions. Finally, the focus of generalization property is different – Fan et al. (2017) focuses on the calibration for binary classification problem, while we focus on the generalization bound that works for general classification and regression problems.

Random-then-Greedy Procedure. Ordered SGD randomly picks a subset of samples and then greedily utilizes a part of the subset, which is related to the random-then-greedy procedure proposed recently in the different topic – the greedy weak learner for gradient boosting (Lu and Mazumder, 2018).

8   Conclusion

We have presented an efficient stochastic first-order method, ordered SGD, for learning an effective predictor in machine learning problems. We have shown that ordered SGD minimizes a new ordered empirical loss $L_q(\theta)$, based on which we have developed the optimization and generalization properties of ordered SGD. The numerical experiments confirmed the effectiveness of our proposed algorithm.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283.

Alain, G., Lamb, A., Sankar, C., Courville, A., and Bengio, Y. (2015). Variance reduction in sgd by distributed importance sampling. *arXiv preprint arXiv:1511.06481*.

Allen-Zhu, Z. (2017). Natasha: Faster non-convex stochastic optimization via strongly non-convex parameter. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 89–97. JMLR. org.

Bartlett, P. L., Foster, D. J., and Telgarsky, M. J. (2017). Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pages 6240–6249.

Bartlett, P. L. and Mendelson, S. (2002). Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482.

Beck, A. and Teboulle, M. (2003). Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175.

Bottou, L., Curtis, F. E., and Nocedal, J. (2018). Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311.

Boyd, S. and Mutapcic, A. (2008). Stochastic subgradient methods. *Lecture Notes for EE364b, Stanford University*.

Davis, D. and Drusvyatskiy, D. (2018). Stochastic subgradient method converges at the rate $O(k^{-1/4})$ on weakly convex functions. *arXiv preprint arXiv:1802.02988*.

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Senior, A., Tucker, P., Yang, K., Le, Q. V., et al. (2012). Large scale distributed deep networks. In *Advances in neural information processing systems*, pages 1223–1231.

Dwork, C., Feldman, V., Hardt, M., Pitassi, T., Reingold, O., and Roth, A. (2015). The reusable holdout: Preserving validity in adaptive data analysis. *Science*, 349(6248):636–638.

Fan, Y., Lyu, S., Ying, Y., and Hu, B. (2017). Learning with average top-k loss. In *Advances in Neural Information Processing Systems*, pages 497–505.

Gopal, S. (2016). Adaptive sampling for SGD by exploiting side information. In *International Conference on Machine Learning*, pages 364–372.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer.

Katharopoulos, A. and Fleuret, F. (2018). Not all samples are created equal: Deep learning with importance sampling. In *International Conference on Machine Learning*, pages 2530–2539.

Kawaguchi, K., Kaelbling, L. P., and Bengio, Y. (2017). Generalization in deep learning. *arXiv preprint arXiv:1710.05468*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Loshchilov, I. and Hutter, F. (2015). Online batch selection for faster training of neural networks. *arXiv preprint arXiv:1511.06343*.

Lu, H. (2017). "relative-continuity" for non-lipschitz non-smooth convex optimization using stochastic (or deterministic) mirror descent. *arXiv preprint arXiv:1710.04718*.

Lu, H., Freund, R., and Mirrokni, V. (2018). Accelerating greedy coordinate descent methods. In *International Conference on Machine Learning*, pages 3263–3272.

Lu, H. and Mazumder, R. (2018). Randomized gradient boosting machine. *arXiv preprint arXiv:1810.10158*.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). *Foundations of machine learning*. MIT press.

Nedic, A. and Lee, S. (2014). On stochastic subgradient mirror-descent algorithm with weighted averaging. *SIAM Journal on Optimization*, 24(1):84–107.

Needell, D., Ward, R., and Srebro, N. (2014). Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. In *Advances in Neural Information Processing Systems*, pages 1017–1025.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch. In *Autodiff Workshop at Conference on Neural Information Processing Systems*.

Rao, R. B., Fung, G., and Rosales, R. (2008). On the dangers of cross-validation. an experimental evaluation. In *Proceedings of the 2008 SIAM international conference on data mining*, pages 588–596. SIAM.

Rockafellar, R. T. and Wets, R. J.-B. (2009). *Variational analysis*, volume 317. Springer Science & Business Media.

Seide, F. and Agarwal, A. (2016). Cntk: Microsoft's open-source deep-learning toolkit. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2135–2135. ACM.

Shalev-Shwartz, S. and Wexler, Y. (2016). Minimizing the maximal loss: How and why. In *International Conference on Machine Learning*, pages 793–801.

Verma, V., Lamb, A., Beckham, C., Najafi, A., Mitliagkas, I., Lopez-Paz, D., and Bengio, Y. (2019). Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning*, pages 6438–6447.

Weston, J., Watkins, C., et al. (1999). Support vector machines for multi-class pattern recognition. In *Esann*, volume 99, pages 219–224.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2017). mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.

Zhang, S. and He, N. (2018). On the convergence rate of stochastic mirror descent for nonsmooth nonconvex optimization. *arXiv preprint arXiv:1806.04781*.

Zhao, P. and Zhang, T. (2015). Stochastic optimization with importance sampling for regularized loss minimization. In *international conference on machine learning*, pages 1–9.

Zhong, Z., Zheng, L., Kang, G., Li, S., and Yang, Y. (2017). Random erasing data augmentation. *arXiv preprint arXiv:1708.04896*.

Kenji Kawaguchi[*], Haihao Lu[*]

Appendix

A Proofs

In Appendix A, we provide complete proofs of the theoretical results.

A.1 Proof of Theorem 1

Proof. We just need to show that $\tilde{g}$ is an unbiased estimator of a sub-gradient of $L_q(\theta)$ at θ^t, namely $\mathbb{E}\tilde{g} \in \partial L_q(\theta^t)$. At first, it holds that

$$\mathbb{E}\tilde{g}^t = \frac{1}{q}\mathbb{E}\sum_{i\in Q}g_i^t + g_R^t = \frac{1}{q}\sum_{i=1}^{n}P(i\in Q)g_i^t + g_R^t = \frac{1}{q}\sum_{j=1}^{n}P((j)\in Q)g_{(j)}^t + g_R^t ~,$$

where $g_i^t \in \partial L_i(\theta^t)$ is a sub-gradient of L_i at θ^t and $g_R^t \in \partial R(\theta^t)$. In the above equality chain, the third equality is simply the definition of expectation, and the last equality is because $((1),(2),\ldots,(n))$ is a permutation of $(1,2,\ldots,n)$.

For any given index j, define $A_j = ((1),(2),\ldots,(j-1))$, then

$$\begin{aligned} P((j)\in Q) &= P\left((j)\in \text{q-argmax}_{i\in S}L_i(\theta)\right) \\ &= P\left((j)\in S \text{ and } S \text{ contains at most } q-1 \text{ items in } A_j\right) \\ &= P\left((j)\in S\right)P\left(S \text{ contains at most } q-1 \text{ items in } A_j|(j)\in S\right) \\ &= P\left((j)\in S\right)\sum_{l=0}^{q-1}P\left(S \text{ contains } l \text{ items in } A_j|(j)\in S\right). \end{aligned} \tag{6}$$

Notice that S is randomly chosen from sample index set $(1,2,\ldots,n)$ without replacement. There are in total $\binom{n}{s}$ different sets S such that $|S|=s$. Among them, there are $\binom{n-1}{s-1}$ different sets S which contains the index (j), thus

$$P\left((j)\in S\right) = \frac{\binom{n-1}{s-1}}{\binom{n}{s}} ~. \tag{7}$$

Given the condition $(j)\in S$, S contains l items in A_j means S contains $s-l-1$ items in $\{(j+1),(j+2)\ldots,(n)\}$, thus there are $\binom{j-1}{l}\binom{n-j}{s-l-1}$ such possible set S, whereby it holds that

$$P\left(S \text{ contains } l \text{ items in } A_j|(j)\in S\right) = \frac{\binom{j-1}{l}\binom{n-j}{s-l-1}}{\binom{n-1}{s-1}} ~. \tag{8}$$

Substituting Equations (7) and (8) into Equation (6), we arrive at

$$P((j)\in T) = \frac{\binom{n-1}{s-1}}{\binom{n}{s}}\sum_{l=0}^{q-1}\frac{\binom{j-1}{l}\binom{n-j}{s-l-1}}{\binom{n-1}{s-1}} = \frac{\sum_{l=0}^{q-1}\binom{j-1}{l}\binom{n-j}{s-l-1}}{\binom{n}{s}} = \gamma_j ~.$$

Therefore,

$$\mathbb{E}\tilde{g}^t = \frac{1}{q}\sum_{j=1}^{n}P((j)\in Q)g_{(j)}^t + g_R^t = \frac{1}{q}\sum_{j=1}^{n}\gamma_j g_{(j)}^t + g_R^t \in \partial L_q(\theta^t) ~,$$

where the last inequality is due to the aditivity of sub-gradient (for both convex and weakly convex function) □

A.2 Proof of Proposition 1

We just need to show that

$$\lim_{j,n\to\infty,j/n=z}\gamma_j = \sum_{l=0}^{q-1}\frac{1}{n}\left(\frac{j}{n}\right)^l\left(\frac{n-j}{n}\right)^{s-l-1}\frac{s!}{l!(s-l-1)!} ~, \tag{9}$$

then we finish the proof by changing variable $z = \frac{j}{n}$.

At first, the Stirling's approximation yields that when n and j are both sufficiently large, it holds that

$$\binom{n}{j} \sim \sqrt{\frac{n}{2\pi j(n-j)}} \frac{n^n}{j^j (n-j)^{n-j}} \cdot \tag{10}$$

Thus,

$$\lim_{j,n \to \infty, j/n=z} \frac{\binom{n-s}{j-1-l}}{\binom{n-1}{j-1}} = \frac{\frac{n^{n-s}}{j^{j-1-l}(n-j)^{n-j-s+1+l}}}{\frac{n^{n-1}}{j^{j-1}(n-j)^{n-j}}} = \frac{j^l(n-j)^{s-l-1}}{n^{s-1}} = \left(\frac{j}{n}\right)^l \left(\frac{n-j}{n}\right)^{s-l-1}, \tag{11}$$

where the first equality utilize Equation (10) and the fact that $s, l, 1$ are negligible in the limit case (except the exponent terms).

On the other hand, it holds by rearranging the factorial numbers that

$$\frac{1}{n} \frac{\binom{n-s}{j-1-l}}{\binom{n-1}{j-1}} \frac{s!}{l!(s-l-1)!} = \frac{\binom{j-1}{l}\binom{n-j}{s-l-1}}{\binom{n}{s}} \cdot \tag{12}$$

Combining Equations (11) and (12) and summing l, we arrive at Equation (9).

By noticing $s > q$, it holds that

$$\begin{aligned}
\frac{d}{dz}\gamma(z) &= \sum_{l=1}^{q-1} l z^{l-1}(1-z)^{s-l-1} \frac{s!}{l!(s-l-1)!} - \sum_{l=0}^{q-1}(s-l-1)z^l(1-z)^{s-l-2}\frac{s!}{l!(s-l-1)!} \\
&= \sum_{l=1}^{q-1} z^{l-1}(1-z)^{s-l-1} \frac{s!}{(l-1)!(s-l-1)!} - \sum_{l=0}^{q-1} z^l(1-z)^{s-l-2}\frac{s!}{l!(s-l-2)!} \\
&= \sum_{l=0}^{q-2} z^l(1-z)^{s-l-2} \frac{s!}{l!(s-l-2)!} - \sum_{l=0}^{q-1} z^l(1-z)^{s-l-2}\frac{s!}{l!(s-l-2)!} \\
&= -z^{q-1}(1-z)^{s-q-1}\frac{s!}{l!(s-l-2)!} \\
&\propto -z^{q-1}(1-z)^{s-q-1}.
\end{aligned}$$

In other word, $1 - \frac{1}{s}\gamma(z)$ is the cumulative of Beta$(q, s-q)$ when $n \to \infty$.

A.3 Proof of Theorem 2

Proof. Notice that $\tilde{g}^t$ is a sub-gradient of $L_Q(\theta^t)$ where $L_Q(\theta^t) = \frac{1}{q}\sum_{i \in Q} L_i(\theta^t) + R(\theta^t)$. Suppose $\tilde{g}^t = \frac{1}{q}\sum_{i \in Q} g_i(\theta^t) + g_R(\theta^t)$ where $g_i(\theta^t)$ is a sub-gradient of $L_i(\theta^t)$ and $g_R(\theta^t)$ is a sub-gradient of $R(\theta^t)$. Then

$$\|\tilde{g}^t\|^2 = \left\| \frac{1}{q}\sum_{i \in Q} g_i(\theta^t) + g_R(\theta^t) \right\|^2 \leq 2\left(\left\| \frac{1}{q}\sum_{i \in Q} g_i(\theta^t) \right\|^2 + \|g_R(\theta^t)\|^2 \right) \leq 2(G_1^2 + G_2^2) . \tag{13}$$

Meanwhile, it follows Theorem 1 that $\tilde{g}^t$ is an unbiased estimator of a sub-gradient of $L_q(\theta^t)$. Together with Equation (13), we obtain the statement (1) by the analysis of convex stochastic sub-gradient descent in Boyd and Mutapcic (2008).

Furthermore, suppose $L_i(\theta) + \frac{\rho}{2}\|\theta\|^2$ is convex for any i, then $L_q(\theta) + \frac{\rho}{2}\|\theta\|^2 = \frac{1}{q}\sum_{j=1}^n \gamma_j \left(L_{(j)}(\theta) + \frac{\rho}{2}\|\theta\|^2\right) + R(\theta)$ is also convex, whereby $L_q(\theta)$ is ρ-weakly convex. We obtain the statement (2) by substituting into Theorem 2.1 in Davis and Drusvyatskiy (2018). $\qquad\square$

A.4 Proof of Theorem 3

Before proving Theorem 3, we first show the following proposition, which gives an upper bound for γ_j:

Proposition 2. *For any $j \in \{1, \dots, n\}$, $\gamma_j \leq \frac{s}{n}$.*

Proof. The value of γ_j is equal to the probability of ordered SGD choosing the j-th sample in the ordered sequence $(L_{(1)}(\theta; \mathcal{D}), \dots, L_{(n)}(\theta; \mathcal{D}))$, which is at most the probability of mini-batch SGD choosing the j-th sample. The probability of mini-batch SGD choosing the j-th sample is $\frac{s}{n}$. ☐

We are now ready to prove Theorem 3 by finding an upper bound on $\sup_{\theta \in \Theta} \mathbb{E}_{(x,y)}[\ell(f(x; \theta), y)] - L_q(\theta; \mathcal{D})$ based on McDiarmid's inequality.

Proof of Theorem 3. Define $\Phi(\mathcal{D}) = \sup_{\theta \in \Theta} \mathbb{E}_{(x,y)}[\ell(f(x; \theta), y)] - L_q(\theta; \mathcal{D})$. In this proof, our objective is to provide the upper bound on $\Phi(\mathcal{D})$ by using McDiarmid's inequality. To apply McDiarmid's inequality to $\Phi(\mathcal{D})$, we first show that $\Phi(\mathcal{D})$ satisfies the remaining condition of McDiarmid's inequality. Let $\mathcal{D}$ and $\mathcal{D}'$ be two datasets differing by exactly one point of an arbitrary index i_0; i.e., $\mathcal{D}_i = \mathcal{D}'_i$ for all $i \neq i_0$ and $\mathcal{D}_{i_0} \neq \mathcal{D}'_{i_0}$. Then, we provide an upper bound on $\Phi(\mathcal{D}') - \Phi(\mathcal{D})$ as follows:

$$\Phi(\mathcal{D}') - \Phi(\mathcal{D}) \leq \sup_{\theta \in \Theta} L_q(\theta; \mathcal{D}) - L_q(\theta; \mathcal{D}').$$

$$= \sup_{\theta \in \Theta} \frac{1}{q} \sum_{j=1}^{n} \gamma_j (L_{(j)}(\theta; \mathcal{D}) - L_{(j)}(\theta; \mathcal{D}'))$$

$$\leq \sup_{\theta \in \Theta} \frac{1}{q} \sum_{j=1}^{n} |\gamma_j| |L_{(j)}(\theta; \mathcal{D}) - L_{(j)}(\theta; \mathcal{D}')|$$

$$\leq \sup_{\theta \in \Theta} \frac{1}{q} \frac{s}{n} \sum_{j=1}^{n} |L_{(j)}(\theta; \mathcal{D}) - L_{(j)}(\theta; \mathcal{D}')|$$

where the first line follows the property of the supremum, $\sup(a) - \sup(b) \leq \sup(a - b)$, the second line follows the definition of L_q, and the last line follows Proposition 2 ($|\gamma_j| \leq \frac{s}{n}$).

We now bound the last term $\sum_{j=1}^{n} |L_{(j)}(\theta; \mathcal{D}) - L_{(j)}(\theta; \mathcal{D}')|$. This requires a careful examination because $|L_{(j)}(\theta; \mathcal{D}) - L_{(j)}(\theta; \mathcal{D}')| \neq 0$ for more than one index j (although $\mathcal{D}$ and $\mathcal{D}'$ differ only by exactly one point). This is because it is possible to have $(j; \mathcal{D}) \neq (j; \mathcal{D}')$ for many indexes j where $(j; \mathcal{D}) = (j)$ in $L_{(j)}(\theta; \mathcal{D})$ and $(j; \mathcal{D}') = (j)$ in $L_{(j)}(\theta; \mathcal{D}')$. To analyze this effect, we now conduct case analysis. Define $l(i; \mathcal{D})$ such that $(j) = i$ where $j = l(i; \mathcal{D})$; i.e., $L_i(\theta; \mathcal{D}) = L_{(l(i;\mathcal{D}))}(\theta; \mathcal{D})$.

Consider the case where $l(i_0; \mathcal{D}') \geq l(i_0; \mathcal{D})$. Let $j_1 = l(i_0; \mathcal{D})$ and $j_2 = l(i_0; \mathcal{D}')$. Then,

$$\sum_{j=1}^{n} |L_{(j)}(\theta; \mathcal{D}) - L_{(j)}(\theta; \mathcal{D}')| = \sum_{j=j_1}^{j_2-1} |L_{(j)}(\theta; \mathcal{D}) - L_{(j)}(\theta; \mathcal{D}')| + |L_{(j_2)}(\theta; \mathcal{D}) - L_{(j_2)}(\theta; \mathcal{D}')|$$

$$= \sum_{j=j_1}^{j_2-1} |L_{(j)}(\theta; \mathcal{D}) - L_{(j+1)}(\theta; \mathcal{D})| + |L_{(j_2)}(\theta; \mathcal{D}) - L_{(j_2)}(\theta; \mathcal{D}')|$$

$$= \sum_{j=j_1}^{j_2-1} (L_{(j)}(\theta; \mathcal{D}) - L_{(j+1)}(\theta; \mathcal{D})) + L_{(j_2)}(\theta; \mathcal{D}) - L_{(j_2)}(\theta; \mathcal{D}')$$

$$= L_{(j_1)}(\theta; \mathcal{D}) - L_{(j_2)}(\theta; \mathcal{D}')$$

$$\leq M,$$

where the first line uses the fact that $j_2 = l(i_0; \mathcal{D}') \geq l(i_0; \mathcal{D}) = j_1$ where i_0 is the index of samples differing in $\mathcal{D}$ and $\mathcal{D}'$. The second line follows the equality $(j; \mathcal{D}') = (j+1; \mathcal{D})$ from j_1 to $j_2 - 1$ in this case. The third line follows the definition of the ordering of the indexes. The fourth line follows the cancellations of the terms from the third line.

Consider the case where $l(i_0; \mathcal{D}') < l(i_0; \mathcal{D})$. Let $j_1 = l(i_0; \mathcal{D}')$ and $j_2 = l(i_0; \mathcal{D})$. Then,

$$
\sum_{j=1}^{n} |L_{(j)}(\theta; \mathcal{D}) - L_{(j)}(\theta; \mathcal{D}')| = |L_{(j_1)}(\theta; \mathcal{D}) - L_{(j_1)}(\theta; \mathcal{D}')| + \sum_{j=j_1+1}^{j_2} |L_{(j)}(\theta; \mathcal{D}) - L_{(j)}(\theta; \mathcal{D}')|
$$

$$
= |L_{(j_1)}(\theta; \mathcal{D}) - L_{(j_1)}(\theta; \mathcal{D}')| + \sum_{j=j_1+1}^{j_2} |L_{(j)}(\theta; \mathcal{D}) - L_{(j-1)}(\theta; \mathcal{D})|
$$

$$
= L_{(j_1)}(\theta; \mathcal{D}) - L_{(j_1)}(\theta; \mathcal{D}') + \sum_{j=j_1+1}^{j_2} (L_{(j)}(\theta; \mathcal{D}) - L_{(j-1)}(\theta; \mathcal{D}))
$$

$$
= L_{(j_1)}(\theta; \mathcal{D}') - L_{(j_2)}(\theta; \mathcal{D})
$$

$$
\leq M.
$$

where the first line uses the fact that $j_1 = l(i_0; \mathcal{D}') < l(i_0; \mathcal{D}) = j_2$ where i_0 is the index of samples differing in $\mathcal{D}$ and $\mathcal{D}'$. The second line follows the equality $(j; \mathcal{D}') = (j-1; \mathcal{D})$ from $j_1 + 1$ to j_2 in this case. The third line follows the definition of the ordering of the indexes. The fourth line follows the cancellations of the terms from the third line.

Therefore, in both cases of $l(i_0; \mathcal{D}') \geq l(i_0; \mathcal{D})$ and $l(i_0; \mathcal{D}') < l(i_0; \mathcal{D})$, we have that

$$
\Phi(\mathcal{D}') - \Phi(\mathcal{D}) \leq \frac{s}{q} \frac{M}{n}.
$$

Similarly, $\Phi(\mathcal{D}) - \Phi(\mathcal{D}') \leq \frac{s}{q} \frac{M}{n}$, and hence $|\Phi(\mathcal{D}) - \Phi(\mathcal{D}')| \leq \frac{s}{q} \frac{M}{n}$. Thus, by McDiarmid's inequality, for any $\delta > 0$, with probability at least $1 - \delta$,

$$
\Phi(\mathcal{D}) \leq \mathbb{E}_{\bar{\mathcal{D}}}[\Phi(\bar{\mathcal{D}})] + \frac{Ms}{q}\sqrt{\frac{\ln(1/\delta)}{2n}}.
$$

Moreover, since

$$
\sum_{i=1}^{n} r_i(\theta; \mathcal{D}) L_i(\theta; \mathcal{D}) = \sum_{j=1}^{n} \gamma_j \sum_{i=1}^{n} \mathbb{1}\{i = (j)\} L_i(\theta; \mathcal{D}) = \sum_{j=1}^{n} \gamma_j L_{(j)}(\theta; \mathcal{D}),
$$

we have that

$$
L_q(\theta; \mathcal{D}) = \frac{1}{q} \sum_{i=1}^{n} r_i(\theta; \mathcal{D}) L_i(\theta; \mathcal{D}) + R(\theta).
$$

Therefore,

$$
\mathbb{E}_{\bar{\mathcal{D}}}[\Phi(\bar{\mathcal{D}})]
$$

$$
= \mathbb{E}_{\bar{\mathcal{D}}}\left[\sup_{\theta \in \Theta} \mathbb{E}_{(\bar{x}', \bar{y}')}[\ell(f(\bar{x}'; \theta), \bar{y}')] - L(\theta; \bar{\mathcal{D}}) + L(\theta; \bar{\mathcal{D}}) - L_q(\theta; \bar{\mathcal{D}}) \right]
$$

$$
\leq \mathbb{E}_{\bar{\mathcal{D}}}\left[\sup_{\theta \in \Theta} \mathbb{E}_{(\bar{x}', \bar{y}')}[\ell(f(\bar{x}'; \theta), \bar{y}')] - L(\theta; \bar{\mathcal{D}}) \right] - \mathcal{Q}_n(\Theta; s, q)
$$

$$
\leq \mathbb{E}_{\bar{\mathcal{D}}, \bar{\mathcal{D}}'}\left[\sup_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} (\ell(f(\bar{x}'_i; \theta), \bar{y}'_i) - \ell(f(\bar{x}_i; \theta), \bar{y}_i)) \right] - \mathcal{Q}_n(\Theta; s, q)
$$

$$
\leq \mathbb{E}_{\xi, \bar{\mathcal{D}}, \bar{\mathcal{D}}'}\left[\sup_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} \xi_i (\ell(f(\bar{x}'_i; \theta), \bar{y}'_i) - \ell(f(\bar{x}_i; \theta), \bar{y}_i)) \right] - \mathcal{Q}_n(\Theta; s, q)
$$

$$
\leq 2\mathfrak{R}_n(\Theta) - \mathcal{Q}_n(\Theta; s, q).
$$

where the third line and the last line follow the subadditivity of supremum, the forth line follows the Jensen's inequality and the convexity of the supremum, the fifth line follows that for each $\xi_i \in \{-1, +1\}$, the distribution

Kenji Kawaguchi[*], Haihao Lu[*]

of each term $\xi_i(\ell(f(\bar{x}'_i; \theta), \bar{y}'_i) - \ell(f(\bar{x}_i; \theta), \bar{y}_i))$ is the distribution of $(\ell(f(\bar{x}'_i; \theta), \bar{y}'_i) - \ell(f(\bar{x}_i; \theta), \bar{y}_i))$ since $\bar{\mathcal{D}}$ and $\bar{\mathcal{D}}'$ are drawn iid with the same distribution. Therefore, for any $\delta > 0$, with probability at least $1 - \delta$,

$$\Phi(\mathcal{D}) \le 2\mathfrak{R}_n(\Theta) - \mathcal{Q}_n(\Theta; s, q) + \frac{Ms}{q}\sqrt{\frac{\ln(1/\delta)}{2n}}.$$

$\square$

B   Additional discussion

The subset Θ in Theorem 3 characterizes the hypothesis space that is $\{x \mapsto f(x; \theta) : \theta \in \Theta\}$. An important subtlety here is that given a parameterized model f, one can apply Theorem 3 to a subset Θ that depends on an algorithm and a distribution (but not directly on a dataset) such as $\Theta = \{\theta \in \mathbb{R}^{d_y} : (\exists \mathcal{D} \in A)[\theta$ is the possible output of ordered SGD given $(f, \mathcal{D})]\}$ where A is a fixed set of the training datasets such that $\mathcal{D} \in A$ with high probability. Thus, even for the exact same model f and problem setting, Theorem 3 might provide non-vacuous bounds for some choices of Θ but not for other choices of Θ.

Moreover, we can easily obtain data-dependent bounds from Theorem 3 by repeatedly applying Theorem 3 to several subsets Θ and taking an union bound. For example, given a sequence $(\Theta_k)_{k \in \mathbb{N}^+}$, by applying Theorem 3 to each Θ_k with $\delta = \delta' \frac{6}{\pi^2 k^2}$ (for each k) and by taking a union bound over all $k \in \mathbb{N}^+$, the following statement holds: for any $\delta' > 0$, with probability at least $1 - \delta'$ over an iid draw of n examples $\mathcal{D} = ((x_i, y_i))_{i=1}^n$, we have that for all $k \in \mathbb{N}^+$ and $\theta \in \Theta_k$,

$$\mathbb{E}_{(x,y)}[\ell(f(x; \theta), y)] \le L_q(\theta; \mathcal{D}) + 2\mathfrak{R}_n(\Theta_k) + \frac{Ms}{q}\sqrt{\frac{\ln(\pi^2 k^2/6\delta')}{2n}} - \mathcal{Q}_n(\Theta_k; s, q).$$

For example, let us choose $\Theta_k = \{\theta \in \mathbb{R}^{d_y} : \|\theta\| \le c_k\}$ with some constants $c_1 < c_2 < \cdots$. Then, when we obtain a $\hat{\theta}_q$ after training based on a particular training dataset $\mathcal{D}$ such that $c_{\bar{k}-1} < \|\hat{\theta}_q\| \le c_{\bar{k}}$ for some $\bar{k}$, we can conclude the following: with probability at least $1 - \delta'$, $\mathbb{E}_{(x,y)}[\ell(f(x; \theta), y)] \le L_q(\hat{\theta}_q; \mathcal{D}) + 2\mathfrak{R}_n(\Theta_{\bar{k}}) + \frac{Ms}{q}\sqrt{\frac{\ln(\pi k^2/6\delta')}{2n}} - \mathcal{Q}_n(\Theta_{\bar{k}}; s, q)$. This is data-dependent in the sense that $\Theta_{\bar{k}}$ is selected in the data-dependent manner from $(\Theta_k)_{k \in \mathbb{N}^+}$. This is in contrast to the fact that as logically indicated in the theorem statement, one cannot directly apply Theorem 3 to a single subset Θ that directly depends on training dataset; e.g., one *cannot* apply Theorem 3 to a singleton set $\hat{\Theta}(\mathcal{D}) = \{\hat{\theta}(\mathcal{D})\}$ where $\hat{\theta}(\mathcal{D})$ is the output of training given $\mathcal{D}$.

C   Additional experimental results and details

C.1   Additional results

Wall-clock time. Table 4 summarises the wall-clock time values (in seconds) of mini-batch SGD and ordered SGD. The wall-clock time was computed with identical, independent, and freed GPUs for fair comparison. The wall-clock time measures the time of the whole computations, including the extra computation of finding a set Q of top-q samples in S in term of loss values. As it can be seen, the extra computation of finding a set Q of top-q samples is generally negligible. Furthermore, for larger scale problems, ordered SGD tends to be faster per epoch because of the computational saving of not using the full mini-batch for the backpropagation computation.

Effect of different learning rates and mini-batch sizes. Figures 5 and 6 show the results with different learning rates and mini-batch sizes. Both use the same setting as that for CIFAR-10 with no data augmentation in others results shown in Table 1 and Figure 3. Figures 5 and 6 consistently show improvement of ordered SGD over mini-batch SGD for all learning rates and mini-batch sizes.

Behaviors with different datasets. Figure 7 shows the behaviors of mini-batch SGD vs ordered SGD. As it can be seen, ordered SGD generally improved mini-batch SGD in terms of test errors. With data argumentation, we also tried linear logistic regression for the Semeion dataset, and obtained the mean test errors of 19.11 for mini-batch SGD and 16.54 for ordered SGD (the standard deviations were 1.48 and 1.24); i.e., ordered SGD
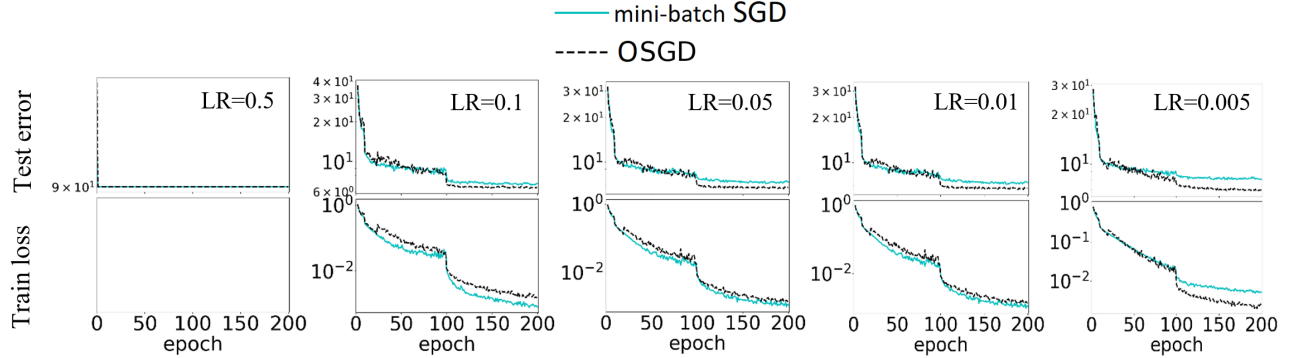
Figure 5: Test error and training loss (in log scales) versus the number of epoch with CIFAR-10 and no data augmentation by using different learning rates (LRs). The plotted values indicate the mean values over 10 random trials. The training loss values of LR=0.5 were 'nan' for both methods.
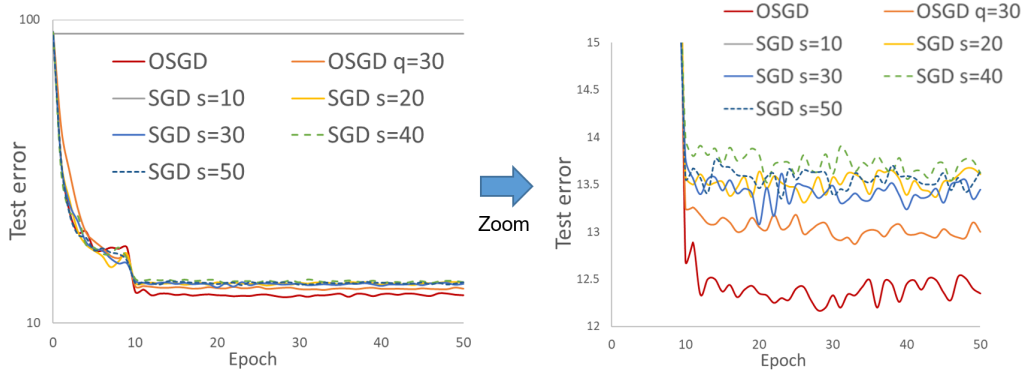

Figure 6: Test error versus the number of epoch with CIFAR-10 and no data augmentation by using different mini-batch sizes s.

improved over mini-batch SGD, but the mean test errors without data-augmentation were better for both mini-batch SGD and ordered SGD. This is because the data augmentation made it difficult to fit the augmented training dataset with linear models.

Effect of different values of q. Figure 8 shows the behaviors of mini-batch SGD vs ordered SGD with different q values. In the figure, label 'ordered SGD' corresponds to ordered SGD with the fixed adaptive rule, and other labels (e.g., 'ordered SGD: $q = 10$') corresponds to ordered SGD with the fixed value of q over the whole training procedure (e.g., with $q = 10$). All experiments in the figure were conducted with data augmentations. PreActResNet18 was used for CIFAR-10, while LeNet was used for other datasets. As it can be seen in Figure 8, ordered SGD generally improved the test errors of mini-batch SGD, even with fixed q values. When the value of q is fixed to be small as in $q = 10$, the small q value can be effective during the latter stage of training (e.g., Figure 8 b) while the training can be inefficient during the initial stage of training (e.g., Figure 8 c).

Results with ordered Adam. Table 5 compares the testing performance of ordered Adam and (standard) Adam for different models and datasets. The table reports the mean and the standard deviation of test errors (i.e., $100 \times$ the average of 0-1 losses on test dataset) over 10 random experiments with different random seeds. The procedures of ordered Adam follow those of Adam except the additional sample strategy (line 3 - 4 of Algorithm 1). Table 5 shows that ordered Adam improved Adam for all settings, except CIFAR-10 with data augmentation. For CIFAR-10 with data augmentation, ordered SGD preformed the best among mini-batch SGD, Adam, ordered SGD, and ordered Adam, as it can be seen in Tables 1 and 5.

Kenji Kawaguchi[*], **Haihao Lu**[*]

Table 4: Average wall-clock time (seconds) per epoch.

| Data Aug	Datasets	Model	mini-batch SGD	ordered SGD	difference
No	Semeion	Logistic model	0.15 (0.01)	0.15 (0.01)	0.00
No	MNIST	Logistic model	7.16 (0.27)	7.32 (0.24)	-0.16
No	Semeion	SVM	0.17 (0.01)	0.17 (0.01)	0.00
No	MNIST	SVM	8.60 (0.31)	8.72 (0.29)	-0.12
No	Semeion	LeNet	0.18 (0.01)	0.18 (0.01)	0.00
No	MNIST	LeNet	9.00 (0.34)	9.12 (0.27)	-0.12
No	KMNIST	LeNet	9.23 (0.33)	9.04 (0.55)	0.19
No	Fashion-MNIST	LeNet	8.56 (0.48)	9.45 (0.31)	-0.90
No	CIFAR-10	PreActResNet18	45.55 (0.47)	43.72 (0.93)	1.82
No	CIFAR-100	PreActResNet18	46.83 (0.90)	43.95 (1.03)	2.89
No	SVHN	PreActResNet18	71.95 (1.40)	66.94 (1.67)	5.01
Yes	Semeion	LeNet	0.28 (0.02)	0.28 (0.02)	0.00
Yes	MNIST	LeNet	14.44 (0.54)	14.77 (0.41)	-0.32
Yes	KMNIST	LeNet	12.17 (0.33)	11.42 (0.29)	0.75
Yes	Fashion-MNIST	LeNet	12.23 (0.40)	12.38 (0.37)	-0.14
Yes	CIFAR-10	PreActResNet18	48.18 (0.58)	46.40 (0.97)	1.78
Yes	CIFAR-100	PreActResNet18	47.37 (0.84)	44.74 (0.91)	2.63
Yes	SVHN	PreActResNet18	72.29 (1.23)	67.95 (1.54)	4.34
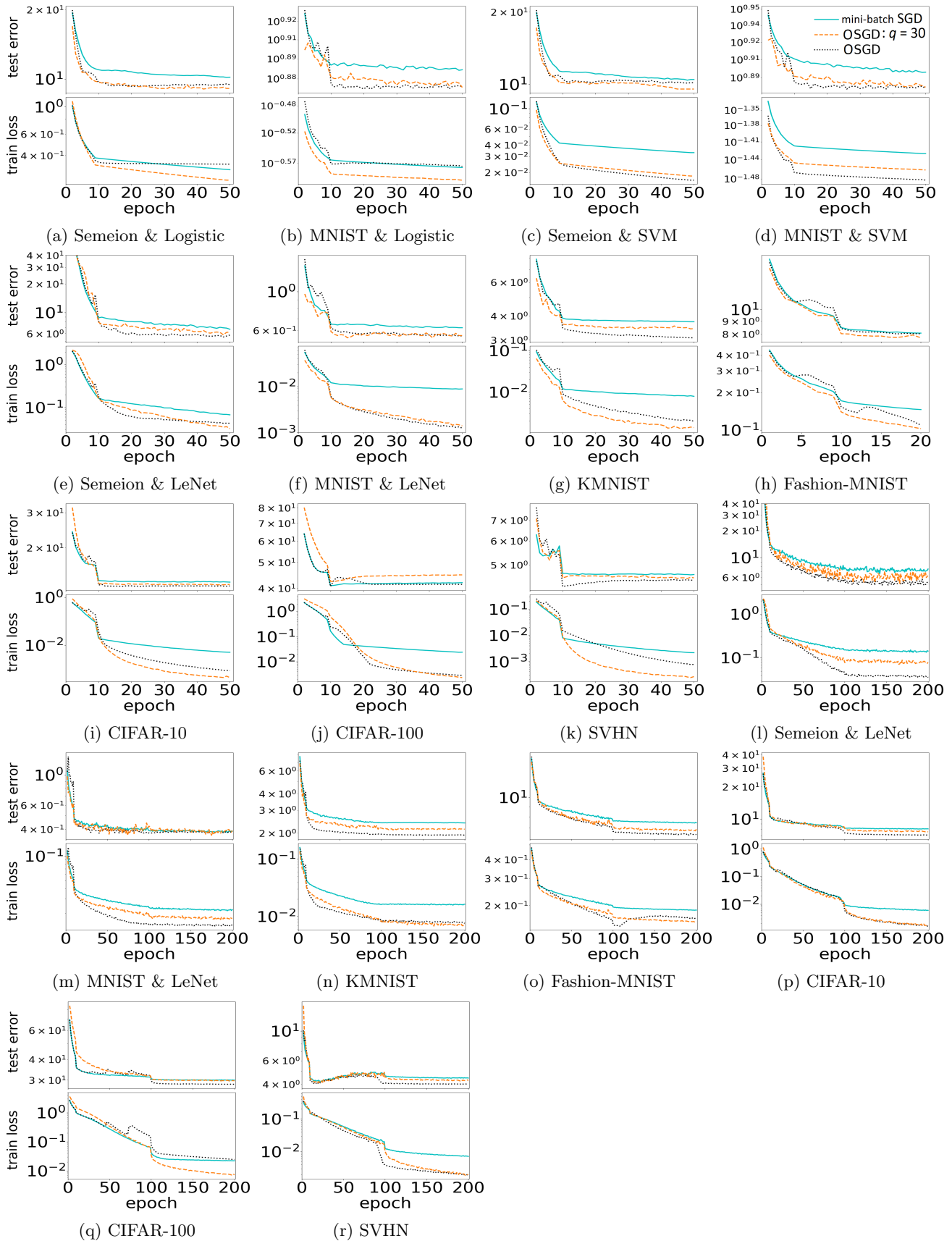
Figure 7: Test error and training loss (in log scales) versus epoch for all experiments with mini-batch SGD and ordered SGD. These are without data augmentation in subfigures (a)-(k), and with data augmentation in subfigures (l)-(r). The plotted values are the mean values over ten random trials.
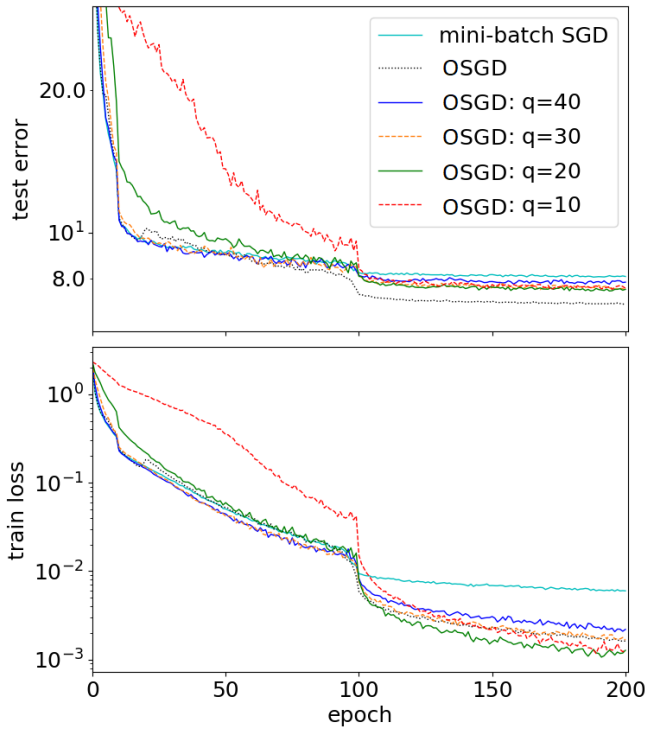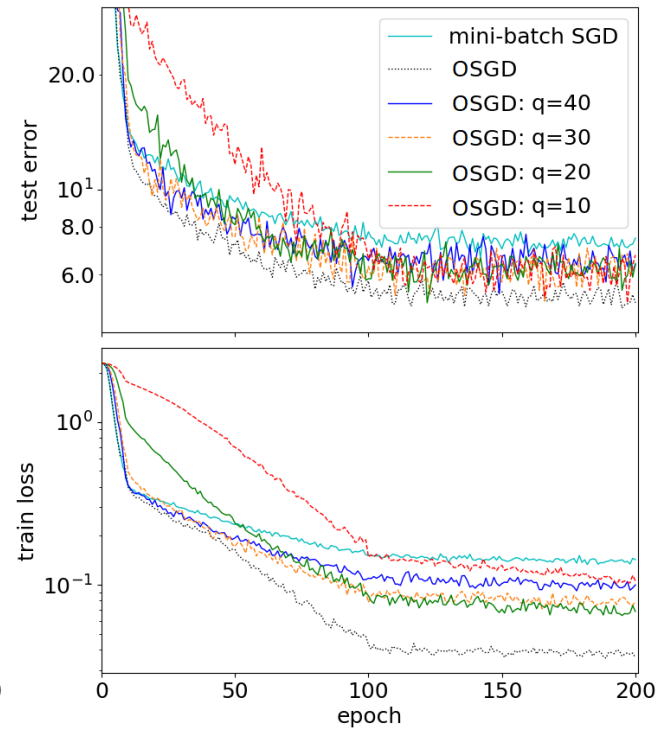
(a) CIFAR-10
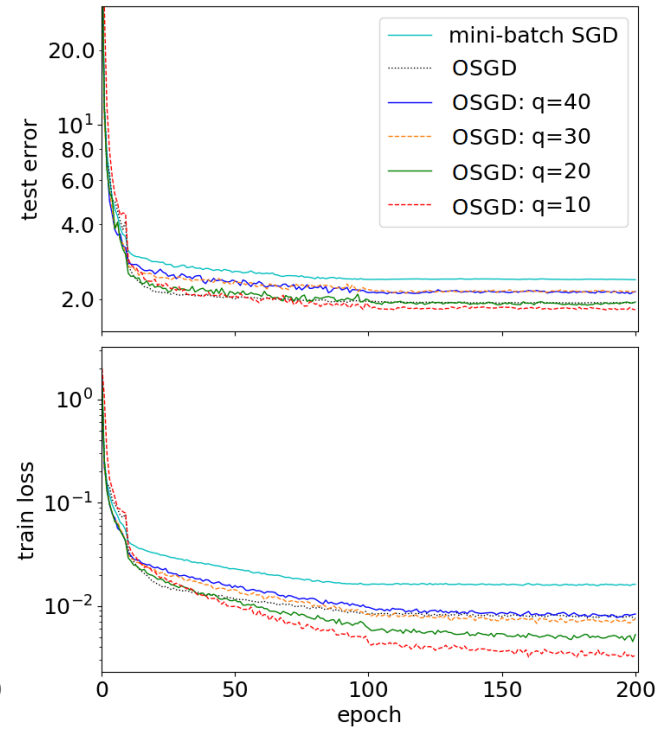
(b) KMNIST
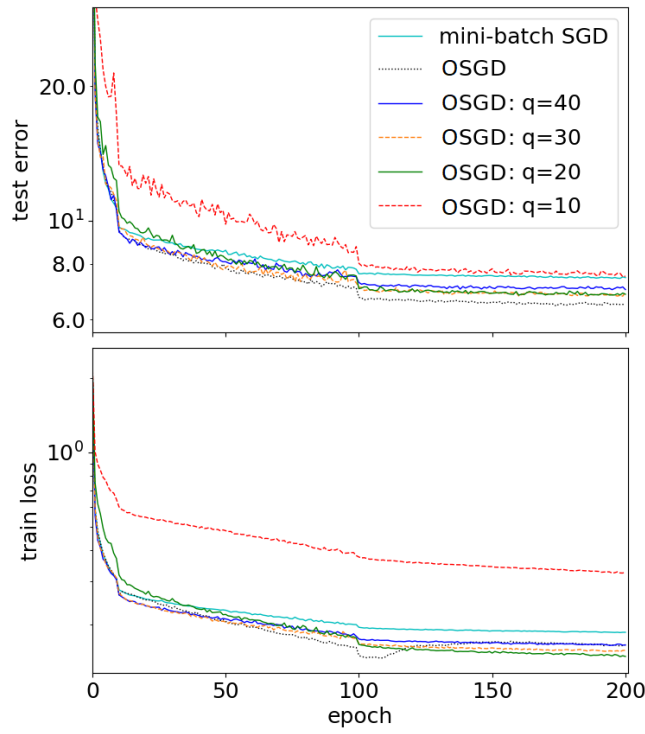
(c) Fashion-MNIST

(d) Semeion

Figure 8: Effect of different values of q.

Table 5: Test errors (%) of Adam and ordered Adam. The last column labeled "Improve" shows relative improvements (%) from Adam to ordered Adam. In the other columns, the numbers indicate the mean test errors (and standard deviations in parentheses) over ten random trials. The first column shows 'No' for no data augmentation, and 'Yes' for data augmentation.

Data Aug	Datasets	Model	Adam	ordered Adam	Improve
No	Semeion	Logistic model	12.12 (0.71)	10.37 (0.77)	14.46
No	MNIST	Logistic model	7.34 (0.03)	7.20 (0.03)	1.97
No	Semeion	SVM	11.45 (0.90)	10.91 (0.86)	4.71
No	MNIST	SVM	7.53 (0.03)	7.43 (0.02)	1.38
No	Semeion	LeNet	6.21 (0.64)	5.75 (0.42)	7.34
No	MNIST	LeNet	0.70 (0.04)	0.63 (0.04)	10.07
No	KMNIST	LeNet	3.14 (0.13)	3.13 (0.14)	0.60
No	Fashion-MNIST	LeNet	7.79 (0.17)	7.79 (0.21)	0.01
No	CIFAR-10	PreActResNet18	13.21 (0.42)	12.98 (0.27)	1.68
No	CIFAR-100	PreActResNet18	45.33 (0.89)	44.42 (0.72)	2.01
No	SVHN	PreActResNet18	4.72 (0.12)	4.64 (0.09)	1.52
Yes	Semeion	LeNet	5.80 (0.85)	5.70 (0.60)	1.74
Yes	MNIST	LeNet	0.45 (0.05)	0.44 (0.02)	3.10
Yes	KMNIST	LeNet	2.01 (0.08)	1.94 (0.16)	3.49
Yes	Fashion-MNIST	LeNet	6.61 (0.14)	6.56 (0.14)	0.82
Yes	CIFAR-10	PreActResNet18	7.92 (0.28)	8.03 (0.13)	-1.39
Yes	CIFAR-100	PreActResNet18	32.24 (0.52)	32.03 (0.52)	0.65
Yes	SVHN	PreActResNet18	4.42 (0.12)	4.19 (0.11)	5.29

C.2   Additional details

For all experiments, mini-batch SGD and ordered SGD (as well as Adam and ordered Adam) were run with the same machine and the same PyTorch codes except a single-line modification:

- `loss = torch.mean(loss)` for mini-batch SGD and Adam

- `loss = torch.mean(torch.topk(loss, min(q, s), sorted=False, dim=0)[0])` for ordered SGD and ordered Adam.

For 2-D illustrations in Figure 1. We used the (binary) cross entropy loss, $s = 100$, and 2 dimensional synthetic datasets with $n = 200$ in Figures 1a–1b and $n = 1000$ in Figures 1c–1d. The artificial neural network (ANN) used in Figures 1c and 1d is a fully-connected feedforward neural network with rectified linear units (ReLUs) and three hidden layers, where each hidden layer contained 20 neurons in Figures 1c and 10 neurons in Figures 1d.

For other numerical results. For mixup and random erasing, we used the same setting as in the corresponding previous papers (Zhong et al., 2017; Verma et al., 2019). For others, we divided the learning rate by 10 at the beginning of 10th epoch for all experiments (with and without data augmentation), and of 100th epoch for those with data augmentation. With $y \in \{1, \ldots, d_y\}$, we used the cross entropy loss $\ell(a, y) = -\log \frac{\exp(a_y)}{\sum_{k'} \exp(a_{k'})}$ for neural networks as well as multinomial logistic models, and a multiclass hinge loss $\ell(a, y) = \sum_{k \neq y} \max(0, 1 + a_k - a_y)$ for SVMs (Weston et al., 1999). For the variant of LeNet, we used the following architecture with five layers (three hidden layers):

1. Input layer

2. Convolutional layer with 64 5×5 filters, followed by max pooling of size of 2 by 2 and ReLU.

3. Convolutional layer with 64 5×5 filters, followed by max pooling of size of 2 by 2 and ReLU.

4. Fully connected layer with 1014 output units, followed by ReLU.

5. Fully connected layer with the number of output units being equal to the number of target classes.