Random Reshuffling with Variance Reduction: New Analysis and Better Rates

Abstract

Virtually all state-of-the-art methods for training supervised machine learning models are variants of SGD, enhanced with a number of additional tricks, such as minibatching, momentum, and adaptive stepsizes. However, one of the most basic questions in the design of successful SGD methods, one that is orthogonal to the aforementioned tricks, is the choice of the *next* training data point to be learning from. Standard variants of SGD employ a *sampling with replacement* strategy, which means that the next training data point is sampled from the entire data set, often independently of all previous samples. While standard SGD is well understood theoretically, virtually all widely used machine learning software is based on *sampling without replacement* as this is often empirically superior. That is, the training data is randomly shuffled/permuted, either only once at the beginning, strategy known as *random shuffling* (Rand-Shuffle), or before every epoch, strategy known as *random reshuffling* (Rand-Reshuffle), and training proceeds in the data order dictated by the shuffling. Rand-Shuffle and Rand-Reshuffle strategies have for a long time remained beyond the reach of theoretical analysis that would satisfactorily explain their success. However, very recently, Mishchenko et al. [2020] provided tight *sublinear* convergence rates through a novel analysis, and showed that these strategies can improve upon standard SGD in certain regimes. Inspired by these results, we seek to further improve the rates of shuffling-based methods. In particular, we show that it is possible to enhance them with a variance reduction mechanism, obtaining *linear* convergence rates. To the best of our knowledge, our linear convergence rates are the best for any method based on sampling without replacement.

1 INTRODUCTION

The main paradigm for training supervised machine learning models—Empirical Risk Minimization (ERM)—is an optimization problem of the finite sum structure

$$\min_{x\in\mathbb{R}^d}\left[f(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x)\right], \tag{1}$$

where $x \in \mathbb{R}^d$ is a vector representing the parameters (model weights, features) of a model we wish to train, n is the number of training data points, and $f_i(x)$ represents the (smooth) loss of the model x on data point i. The goal of ERM is to train a model whose average loss on the training data is minimized. This abstraction allows to encode virtually all supervised models trained in practice, including linear and logistic regression, and neural networks.

The gigantic size of modern training data sets necessary to train models with good generalization poses severe issues for the designers of methods for solving (1). Over the last decade, stochastic first-order methods have emerged as the methods of choice, and for this reason, their importance in machine learning remains exceptionally high [Bottou et al., 2018]. Of these, stochastic gradient descent (SGD) is perhaps the best known, but also the most basic. SGD has a long history [Robbins and Monro, 1951, Bertsekas and Tsitsiklis, 1996] and is therefore well-studied and well-understood [Rakhlin et al., 2012, Hardt et al., 2016, Drori and Shamir, 2019, Gower et al., 2019, Nguyen et al., 2020].

Training data order. Standard and even variance-reduced variants of SGD employ a *sampling with replacement* strategy [Gorbunov et al., 2020], which means that the next training data point in each epoch is sampled from the entire data set, independently of all previous samples. However, virtually all widely used machine learning software is based on *sampling without replacement* as this is often empirically superior [Bottou, 2009, Recht and Ré, 2013], and therefore acts as the de-facto default sampling mechanism in deep learning [Bengio, 2012, Sun, 2020]. With this latter strategy, in each epoch we sample each training data exactly

Submitted to the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023). **To be used for reviewing only**.

once, and this can be performed by generating a random permutation of the training data.

There are three commonly used variants of sampling without replacement.

(i) In the first, which we call *deterministic shuffling* (Det-Shuffle) in this paper, the training data is processed in some natural order in a cyclic manner. That is, a deterministic permutation is used throughout the entire training process. This idea is the basis of the Cyclic-GD method [Luo, 1991, Grippo, 1994]. While this strategy is not effective in practice, it is perhaps the simplest strategy conceptually, and has been studied repeatedly. However, it is notoriously difficult to obtain good guarantees for it.

(ii) In the second variant, which we call *random shuffling* (Rand-Shuffle) in this paper[1], the training data is instead shuffled/permuted randomly. This is done only once, before the start of the training process, and the selection of training data then follows a cyclic pattern dictated by this single random permutation [Nedić and Bertsekas, 2001]. The purpose of this procedure is to break the potentially adversarial default ordering of the data that could negatively affect training speed. Almost no non-trivial analyses exist for this method [Mishchenko et al., 2020]. This strategy works very well in practice.

(iii) In the third variant, known as *random reshuffling* (Rand-Reshuffle), the training data is randomly reshuffled before the start of each epoch. This is perhaps the most common and relatively most studied approach. Its empirical performance is, however, often very similar to Rand-Shuffle, and the current best theoretical bounds for both are the same [Mishchenko et al., 2020].

Difficulties with analyzing shuffling-based methods. The main difficulty in analyzing methods based on sampling without replacement is that each gradient step within an epoch is *biased*, and performing a sharp analysis of methods based on biased estimators is notoriously difficult. While Cyclic-GD was studied already a few decades ago [Mangasarian and Solodov, 1994, Bertsekas and Tsitsiklis, 2000], convergence rates were established relatively recently [Li et al., 2019, Ying et al., 2019, Gürbüzbalaban et al., 2019, Nguyen et al., 2020]. For the Rand-Shuffle method, the situation is more complicated, and non-vacuous theoretical analyses were only performed recently [Safran and Shamir, 2020, Rajput et al., 2020]. Rand-Reshuffle is well understood for twice-smooth [Gürbüzbalaban et al., 2019, Haochen and Sra, 2019] and smooth [Nagaraj et al., 2019] objectives. Moreover, lower bounds for Rand-Reshuffle and similar methods were also recently established [Safran and Shamir, 2020, Rajput et al., 2020]. Mishchenko et al. [2020] recently performed an in-depth analysis of Det-Shuffle, Rand-Shuffle and Rand-Reshuffle with novel and simpler proof techniques, leading to improved and new convergence rates. Their rate for Rand-Shuffle, for example, tightly matches the lower bound of Safran and Shamir [2020] in the case when each f_i is strongly convex. Further, Rand-Reshuffle can be accelerated [Gürbüzbalaban et al., 2019], and for small constant step-sizes, the neighborhood of solution can be controlled [Sayed, 2014]. However, despite these advances, Rand-Reshuffle and related method described above still suffer from the same problem as SGD, i.e., we do not have variants that would have a fast linear convergence rate to the exact minimizer.

Variance reduction. Despite its simplicity and elegance, SGD has a significant disadvantage: the variance of naive stochastic gradient estimators of the true gradient remains high throughout the training process, which causes issues with convergence. When a constant learning rate is used in the smooth and strongly convex regime, SGD converges linearly to a neighborhood of the optimal solution of size proportional to the learning rate and to the variance of the stochastic gradients at the optimum [Gower et al., 2020]. While a small or a decaying learning schedule restores convergence, the convergence speed suffers as a result. Fortunately, there is a remedy for this ailment: *variance-reduction* (VR) [Johnson and Zhang, 2013]. The purpose of VR mechanisms is to steer away from the naive gradient estimators. Instead, VR mechanisms iteratively construct and apply a gradient estimator whose variance would eventually vanish. This allows for larger learning rates to be used safely, which accelerates training. Among the early VR-empowered SGD methods belong SAG [Roux et al., 2012], SVRG [Johnson and Zhang, 2013], SAGA [Defazio et al., 2014a], and Finito [Defazio et al., 2014b]. For a recent survey of VR methods, see [Gower et al., 2020].

Related work. Some cyclic and random reshuffling versions of variance-reduced methods were shown to obtain linear convergence. Incremental Average Gradient (IAG)—a cyclic version of the famous SAG method—was analyzed by Gürbüzbalaban et al. [2017]. Based on this, the Doubly Incremental Average Gradient (DIAG) method was introduced, and it has a significantly better rate if each f_i is strongly convex [Mokhtari et al., 2018]. A linear rate for Cyclic-SAGA was established by Park and Ryu [2020]. The first analysis of Rand-Reshuffle with variance reduction was done by Ying et al. [2020]. Firstly, they establish a linear rate for SAGA under random reshuffling, and then they introduce a new method called Amortized Variance-Reduced Gradient (AVRG), which is similar to SAGA. SVRG using Rand-Reshuffle was introduced by Shamir [2016], and their theoretical analysis was conducted for the Least Squares problem. The promising result of Prox-DFinito is introduced in Huang et al. [2021] for the composite optimization problem.

[1]This method is called "shuffle once" in some papers.

2 APPROACH AND CONTRIBUTIONS

Let us now briefly outline our approach and key contributions.

2.1 CONTROLLED LINEAR PERTURBATIONS

In the design of our methods we employ a simple but powerful tool: the idea of introducing a sequence of carefully crafted reformulations of the original finite sum problem, and applying vanilla shuffling-based methods on these reformulations instead of the original formulation. As the sequence is designed to have progressively better conditioning properties, our methods will behave progressively better as well, and this is why this result in variance reduced shuffling methods.

The main idea is to perturb the objective function with zero written as the average of n nonzero linear functions. This perturbation is performed at the beginning of each epoch, and stays fixed within each epoch. Let us consider the finite sum problem (1) and vectors $a_t^i, \dots, a_t^n \in \mathbb{R}^d$ summing up to zero: $\sum_{i=1}^n a_t^i = 0$. Let $a^t = (a_t^i, \dots, a_t^n)$. Adding this *structured* zero to f, we reformulate problem (1) into the equivalent form

$$f(x) = \frac{1}{n}\sum_{i=1}^n f_i(x) = \frac{1}{n}\sum_{i=1}^n \left(f_i(x) + \langle a_t^i, x\rangle\right) := \frac{1}{n}\sum_{i=1}^n f_i^t(x), \tag{2}$$

where $f_i^t(x) := f_i(x) + \langle a_t^i, x\rangle$. Note that

$$\nabla f_i^t(x) = \nabla f_i(x) + a_t^i. \tag{3}$$

Next, we establish a simple but important property of this reformulation.

Proposition 1. *Assume that each f_i is μ-strongly convex (resp. convex) and L-smooth. Then f_i^t is μ-strongly convex (resp. convex) and L-smooth.*

In our methods, the vectors $a_t^1, \dots, a_t^n$ depend on two objects:

- a *control vector* $y_t \in \mathbb{R}^d$, which is updated at the start of each epoch,
- the *permutation* $\pi = \{\pi_0, \pi_1, \dots, \pi_{n-1}\}$ chosen at the beginning of the current epoch.

In particular, we choose

$$a_t^i := -\nabla f_{\pi_i}(y_t) + \nabla f(y_t). \tag{4}$$

Note that by plugging (4) into (3), the gradient of $f_{\pi_i}^t$ at $x \in \mathbb{R}^d$ is given by

$$g_t^i(x, y_t) := \nabla f_{\pi_i}(x) - \nabla f_{\pi_i}(y_t) + \nabla f(y_t). \tag{5}$$

At the start of each epoch, the control vector y_t is set to the latest iterate x_t.

Algorithm 1 Algorithms Det-Shuffle, Rand-Shuffle, Rand-Reshuffle

Input: Stepsize $\gamma > 0$, initial iterate $x_0 \in \mathbb{R}^d$, number of epochs T
Option Det-Shuffle**:** Choose a deterministic permutation $\{\pi_0, \dots, \pi_{n-1}\}$ of $\{1, \dots, n\}$
Option Rand-Shuffle**:** Choose a random permutation $\{\pi_0, \dots, \pi_{n-1}\}$ of $\{1, \dots, n\}$
for $t = 0, 1, \dots T-1$ **do**
  Option Rand-Reshuffle**:** Choose a random permutation $\{\pi_0, \dots, \pi_{n-1}\}$ of $\{1, \dots, n\}$
  $x_t^0 = x_t$, $y_t = x_t$
  for $i = 0, \dots, n-1$ **do**
    $g_t^i(x_t^i, y_t) = \nabla f_{\pi_i}(x_t^i) - \nabla f_{\pi_i}(y_t) + \nabla f(y_t)$
    $x_t^{i+1} = x_t^i - \gamma g_t^i(x_t^i, y_t)$
  end for
  $x_{t+1} = x_t^n$
end for

2.2 NEW ALGORITHMS: IMPROVEMENT OF SHUFFLING BASED METHODS

Our key proposal is to run *standard* Det-Shuffle, Rand-Shuffle and Rand-Reshuffle methods, for example as described in [Mishchenko et al., 2020], but in each epoch to apply them to the current reformulated problem

$$\min_{x\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^n f_i^t(x).$$

This leads to our variance-reduced algorithms, all described compactly in Algorithm 1. Hoping that this will not cause confusion, we do not give the methods a different name.

- Note that as mentioned in the introduction, in Det-Shuffle we only use a single deterministic permutation at the start of the method. The steps are then performed incrementally through all data, in the same order in each epoch.
- In contrast, in Rand-Shuffle we shuffle the data points randomly instead, but otherwise proceed as in Det-Shuffle, using this one permutation in all subsequent epochs.
- Finally, Rand-Reshuffle is similar to Rand-Shuffle, with the exception that a new permutation is resampled at the start of each epoch.[2]

Besides Algorithm 1, we also propose a generalized version of Rand-Reshuffle (Algorithm 2), which differs from Rand-Reshuffle in that at the end of each epoch we flip a biased

[2]Note that Rand-Reshuffle can be seen as a version of SVRG in which the number of inner steps m is equal to n, and in which sampling *without* replacement is used. Johnson and Zhang [2013] remarked that $m = \mathcal{O}(n)$ works well in practice, but a theoretical analysis of this was not provided.

coin to decide whether to update the control vector y_t or not. While in Rand-Reshuffle the control vector y_{t+1} is updated to the latest iterate x_{t+1}, in Algorithm 2 we use the previous point x_t. We do this as it slightly simplified the analysis. However, it makes sense to use the newest point x_{t+1} instead of x_t to update the control vector in practice. This method is described in the appendix only.

2.3 ANALYSIS TECHNIQUE: THE BASIC IDEA

Since in view of Proposition 1 the reformulated problem satisfies all assumptions of the original problem, in a single epoch it is possible to apply results that hold for vanilla Det-Shuffle, Rand-Shuffle and Rand-Reshuffle methods – variants that are not variance-reduced. In particular, we rely on some results of Mishchenko et al. [2020], and complement them with new analysis that handles the changing nature of the reformulations through the change in the control vectors $\{y_t\}$.

In particular, a key insight of our paper is the observation that by updating the control vector, we can control the variance of shuffling based methods.[3]

We are now ready to formulate the core lemma of our work.

Lemma 1. *Assume that each f_i is L-smooth and convex. If we apply the linear perturbation reformulation* (2) *using vectors of the form* (4)*, then the gradient variance of the reformulated problem at the optimum x_* can be bounded via the distance of the control vector y_t to x_* as follows:*

$$(\sigma_*^t)^2 := \frac{1}{n}\sum_{i=1}^{n}\left\|\nabla f_i^t(x_*)\right\|^2 \leq 4L^2\|y_t - x_*\|^2. \qquad (6)$$

2.4 COMPLEXITY RESULTS

Our theory leads to improved rates for shuffling-based methods using all three sampling strategies: Det-Shuffle, Rand-Shuffle and Rand-Reshuffle. We provide theoretical guaranties in Section 3; a summary is presented in Table 1.

◇ **Strongly convex case.** If f is strongly convex, we obtain $\mathcal{O}\left(\kappa^{3/2}\log 1/\varepsilon\right)$ iteration (epoch-by-epoch) complexity for Rand-Reshuffle, where κ is the condition number. This rate is better than the $\mathcal{O}\left(\kappa^2\log 1/\varepsilon\right)$ rate of RR-SAGA and AVRG introduced by Ying et al. [2020]. Moreover, if $n > \mathcal{O}(\kappa)$, we improve this rate for Rand-Reshuffle and get $\mathcal{O}\left(\kappa\log 1/\varepsilon\right)$ complexity. If each f_i is strongly convex and the number of functions is sufficiently large (Theorem 3), then the rate of Rand-Reshuffle can be further improved to $\mathcal{O}(\kappa\sqrt{\kappa/n}\log 1/\varepsilon)$. For Det-Shuffle we prove similar convergence results under the assumption of strong convexity of f.

[3] While this was known for methods based on sampling with replacement, this is a new observation for methods based on sampling without replacement, and our control strategy.

The iteration complexity of this method is $\mathcal{O}\left(\kappa^{3/2}\log 1/\varepsilon\right)$, which is noticeably better than the $\mathcal{O}\left(n\kappa^2\log 1/\varepsilon\right)$ rate of IAG [Gürbüzbalaban et al., 2017]. Furthermore, it is better than the $\mathcal{O}\left(\kappa^2\log 1/\varepsilon\right)$ rate of Cyclic-SAGA [Park and Ryu, 2020]. It is worth mentioning that Mokhtari et al. [2018] obtain a better complexity, $\mathcal{O}\left(\kappa\log 1/\varepsilon\right)$, for their DIAG method. However, their analysis requires much stricter assumption.

◇ **Convex case.** In the general convex setting we give the first analysis and convergence guarantees for Det-Shuffle, Rand-Shuffle, and Rand-Reshuffle. After applying variance reduction, we obtain fast convergence to the exact solution. As expected, these methods have the sublinear rate $\mathcal{O}(\frac{1}{\varepsilon})$ in an ergodic sense.

2.5 SHUFFLING-BASED VARIANTS OF VARIANCE REDUCED METHODS.

While, as we argue, our methods should be seen as improvements over existing shuffling-based methods via variance reduction, it is possible to alternatively see them as shuffling-based variants of variance reduced methods. However, when seen that way, we do not observe an improvement in complexity. The reason for this is that there is a large gap in our understanding of shuffling based methods, especially for variance reduced variants, which does not yet allow for theoretical speedups compared to their sampling-with-replacement cousins. For example, from the latter viewpoint, and to the best of our knowledge, we provide the first convergence analysis of SVRG under random reshuffling. However, the rate of classical variance reduced methods, such as SVRG, is still superior in some regimes.

3 MAIN THEORETICAL RESULTS

Having described the methods and the idea of controlled linear perturbations, we are ready to proceed to the formal statement of our convergence results.

3.1 ASSUMPTIONS AND NOTATION

Before introducing our convergence results, let us first formulate the definitions and assumptions we use throughout the work. Function $f : \mathbb{R}^d \to \mathbb{R}$ is L-smooth if

$$f(y) \leq f(x) + \langle\nabla f(x), y - x\rangle + \frac{L}{2}\|y - x\|^2 \quad \forall x, y \in \mathbb{R}^d, \qquad (7)$$

convex if

$$f(x) + \langle\nabla f(x), y - x\rangle \leq f(y) \quad \forall x, y \in \mathbb{R}^d, \qquad (8)$$

and μ-strongly convex if

$$f(x) + \langle\nabla f(x), y - x\rangle + \frac{\mu}{2}\|y - x\|^2 \leq f(y) \quad \forall x, y \in \mathbb{R}^d. \qquad (9)$$

Table 1: Complexity of shuffling based methods (in all expressions we ignore constant terms).

Algorithm	**μ-strongly convex f_i**	**μ-strongly** convex f	**convex** f	**non-convex** f	**memory**	**reference**
RR-SAGA	–	$\kappa^2 \log 1/\epsilon$	–	–	dn	Ying et al. [2020]
AVRG	–	$\kappa^2 \log 1/\epsilon$	–	–	d	Ying et al. [2020]
Rand-Shuffle Rand-Reshuffle	$\kappa\sqrt{\frac{\kappa}{n}} \log 1/\epsilon$ (1) $\kappa\sqrt{\kappa} \log 1/\epsilon$ (2)	$\kappa \log 1/\epsilon$ (1) $\kappa\sqrt{\kappa} \log 1/\epsilon$ (2)	L/ε	L/ϵ^2 (5)	d	this paper
Prox-DFinito	$\kappa \log 1/\epsilon$	–	L^2/ϵ	–	dn	Huang et al. [2021]
Cyclic-SAGA	$\kappa^2 \log 1/\epsilon$	–	–	–	dn	Park and Ryu [2020]
IAG (3)	–	$n\kappa^2 \log 1/\epsilon$	–	–	dn	Gürbüzbalaban et al. [2017]
DIAG (4)	$\kappa \log 1/\epsilon$	–	–	–	dn	Mokhtari et al. [2018]
Det-Shuffle	–	$\kappa\sqrt{\kappa} \log 1/\epsilon$	L/ϵ	–	d	this paper

(1) Big data regime.
(2) General regime.
(4) Cyclic version of the Stochastic Average Gradient (SAG) method, which was the original inspiration for SAG.
(3) Cyclic version of the Finito algorithm.
(5) The result is applied to Rand-Reshuffle

The Bregman divergence with respect to f is the mapping $D_f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ defined as follows:

$$D_f(x, y) := f(x) - f(y) - \langle \nabla f(y), x - y \rangle. \tag{10}$$

Note that if $y = x_*$, where x_* is a minimum of f, then $D_f(x, x_*) = f(x) - f(x_*)$.

Lastly, we define an object that plays the key role in our analysis.

Definition 3.1 (Variance at optimum)**.** *Gradient variance at optimum is the quantity*

$$\sigma_*^2 := \frac{1}{n} \sum_{i=1}^{n} \|\nabla f_i(x_*)\|^2 . \tag{11}$$

This quantity is used in several recent papers on stochastic gradient-type methods. Particularly, it is a version of gradient noise introduced in Gower et al. [2019] for finite sum problems.

For all theorems in this paper the following assumption is used.

Assumption 1. *The objective f and the individual losses $f_1, \ldots, f_n$ are all L-smooth. We also assume the existence of a minimizer $x_* \in \mathbb{R}^d$.*

This assumption is classical in the literature, and it is necessary for us to get convergence results for all the methods described above.

3.2 CONVERGENCE ANALYSIS OF RAND-SHUFFLE AND RAND-RESHUFFLE

We provide two different rates in the strongly convex case. Let $\kappa := L/\mu$.

Theorem 1 (Strongly convex case: f)**.** *Suppose that each f_i is convex, f is μ-strongly convex, and Assumption 1 holds. If the stepsize satisfies $0 < \gamma \leq (2\sqrt{2}Ln\sqrt{\kappa})^{-1}$, the iterates generated by* Rand-Shuffle *and* Rand-Reshuffle *satisfy*

$$\mathbb{E}\left[\|x_T - x_*\|^2\right] \leq \left(1 - \tfrac{\gamma n \mu}{2}\right)^T \|x_0 - x_*\|^2.$$

This means that the iteration complexity of these methods is $T = \mathcal{O}\left(\kappa\sqrt{\kappa} \log 1/\varepsilon\right)$.

If we are in the big data regime characterized by the inequality $n > \mathcal{O}(\kappa)$, then we can use a larger step-size, which leads to an improved rate. This is captured by our next theorem.

Theorem 2 (Strongly convex case: f)**.** *Suppose that each f_i is convex, f is μ-strongly convex and Assumption 1 holds. Additionally assume we are in the "big data" regime characterized by $n \geq 2\kappa/(1 - \frac{1}{\sqrt{2}\kappa})$. Then provided the stepsize satisfies $\gamma \leq 1/\sqrt{2}Ln$, the iterates generated by* Rand-Shuffle *and* Rand-Reshuffle *satisfy*

$$\mathbb{E}\left[\|x_T - x_*\|^2\right] \leq \left(1 - \tfrac{\gamma n \mu}{2}\right)^T \|x_0 - x_*\|^2.$$

This means that the iteration complexity of these methods is $T = \mathcal{O}\left(\kappa \log 1/\varepsilon\right)$.

As we shall see next, we obtain an even better rate in the case when each function f_i is strongly convex.

Theorem 3 (Strongly convex case: f_i)**.** *Suppose that the functions $f_1, \ldots, f_n$ are μ-strongly convex and Assumption 1 holds. Fix constant $0 < \delta < 1$. If the stepsize satisfies $\gamma \leq \delta/L\sqrt{2n\kappa}$, and if number of functions is sufficiently big, $n > \log(1-\delta^2)/\log(1-\gamma\mu)$, then the iterates generated by* Rand-Shuffle *and* Rand-Reshuffle *satisfy*

$$\mathbb{E}\left[\|x_T - x_*\|^2\right] \leq \left((1 - \gamma\mu)^n + \delta^2\right)^T \|x_0 - x_*\|^2.$$

If we further assume that $\delta^2 \leq (1 - \gamma\mu)^{n/2}\left(1 - (1 - \gamma\mu)^{n/2}\right)$, then the iteration complexity of these methods is $T = \mathcal{O}\left(\kappa\sqrt{\kappa/n} \log 1/\varepsilon\right)$.

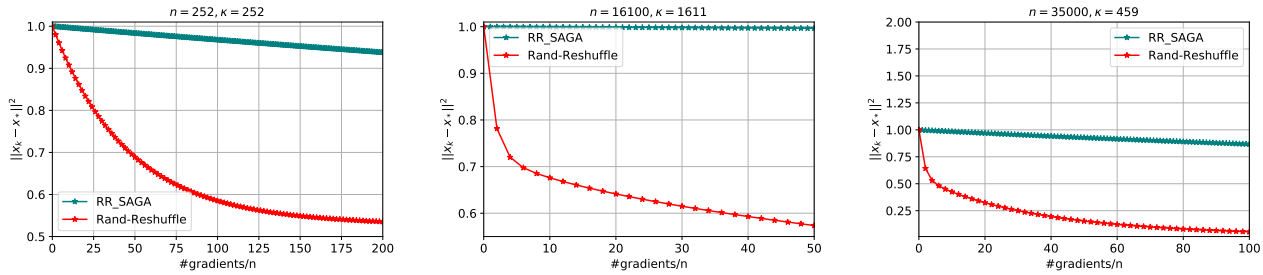


Figure 1: Comparison of Rand-Reshuffle and RR-SAGA with theoretical stepsizes on `bodyfat`, `a7a`, and `ijcnn1` datasets (from left to right).

In our work we provide the first bounds for SVRG under random reshuffling without strong convexity.

Theorem 4 (Convex case)**.** *Suppose the functions $f_1, f_2, \ldots, f_n$ are convex and Assumption 1 holds. Then for* Rand-Shuffle *and* Rand-Reshuffle *with stepsize $\gamma \leq 1/\sqrt{2}Ln$, the average iterate $\hat{x}_T := \frac{1}{T}\sum_{t=1}^{T} x_t$ satisfies*

$$\mathbb{E}\left[f\left(\hat{x}_T\right)\right] - f\left(x_*\right) \leq \frac{3\|x_0 - x_*\|^2}{2\gamma n T}.$$

This means that the iteration complexity of these methods is $T = \mathcal{O}\left(\frac{L\|x_0 - x_*\|^2}{\varepsilon}\right)$.

We also obtained first convergence result for Rand-Reshuffle in the non-convex case.

Theorem 5 (General non-convex case)**.** *Suppose that Assumption 1 holds. Then for Algorithm* Rand-Reshuffle *run for T epochs with a stepsize $\gamma \leq \frac{1}{2Ln}$ we have*

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\left[\|\nabla f\left(x_t\right)\|^2\right] \leq \frac{4(f(x_0) - f_*)}{\gamma n T}.$$

Choose $\gamma = \frac{1}{2nL}$. Then the mean of gradient norms satisfies $\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\left[\|\nabla f\left(x_t\right)\|^2\right] \leq \varepsilon^2$ provided the number of iterations satisfies $T = \mathcal{O}\left(\frac{8\delta_0 L}{\varepsilon^2}\right)$.

Theorem 6 (Polyak-Łojasiewicz condition)**.** *Suppose that Assumption 1 holds and f satisfies the Polyak-Łojasiewicz inequality with $\mu > 0$, i.e., $\|\nabla f(x)\|^2 \geq 2\mu(f(x) - f_*)$ for any $x \in \mathbb{R}^d$. Then for Algorithm* Rand-Reshuffle *run for T epochs with a stepsize $\gamma \leq \frac{1}{2Ln}$ we have*

$$\mathbb{E}\left[f(x_T) - f_*\right] \leq \left(1 - \frac{\gamma\mu n}{2}\right)^T \left(f(x_0) - f_*\right),$$

then the relative error satisfies $\frac{\mathbb{E}[f(x_T) - f_*]}{f(x_0) - f_*} \leq \varepsilon$ provided the number of iterations satisfies $T = \mathcal{O}(\kappa \log \frac{1}{\varepsilon})$.

3.3 CONVERGENCE ANALYSIS OF DET-SHUFFLE

In this section we present results for Det-Shuffle. They are very similar to the previous bounds. However, the lack of randomization does not allow us to improve convergence in the big data regime.

Theorem 7 (Strongly convex case: f)**.** *Suppose that each f_i is convex function, f is μ-strongly convex function, and Assumption 1 holds. If the stepsize satisfies $\gamma \leq 1/4Ln\sqrt{\kappa}$, the iterates generated by* Det-Shuffle *satisfy*

$$\|x_T - x_*\|^2 \leq \left(1 - \frac{\gamma n \mu}{2}\right)^T \|x_0 - x_*\|^2.$$

This means that the iteration complexity of this method is $T = \mathcal{O}\left(\kappa\sqrt{\kappa}\log 1/\varepsilon\right)$.

Note that this is the same rate as that of Rand-Shuffle and Rand-Reshuffle.

Our rate for Det-Shuffle is better than the rate of Cyclic-SAGA [Park and Ryu, 2020]. We remark that the convergence rate of DIAG [Mokhtari et al., 2018] is better still; however, their result requires strong convexity of each f_i.

Similarly, we can establish convergence results for Det-Shuffle in the convex case.

Theorem 8 (Convex case)**.** *Suppose the functions $f_1, f_2, \ldots, f_n$ are convex and Assumption 1 holds. If the stepsize satisfies $\gamma \leq 1/2\sqrt{2}Ln$, the average iterate $\hat{x}_T := \frac{1}{T}\sum_{j=1}^{T} x_j$ generated by* Det-Shuffle *satisfies*

$$\mathbb{E}\left[f\left(\hat{x}_T\right)\right] - f\left(x_*\right) \leq \frac{2\|x_0 - x_*\|^2}{\gamma n T}.$$

This means that the iteration complexity of this method is $T = \mathcal{O}\left(\frac{L\|x_0 - x_*\|^2}{\varepsilon}\right)$.

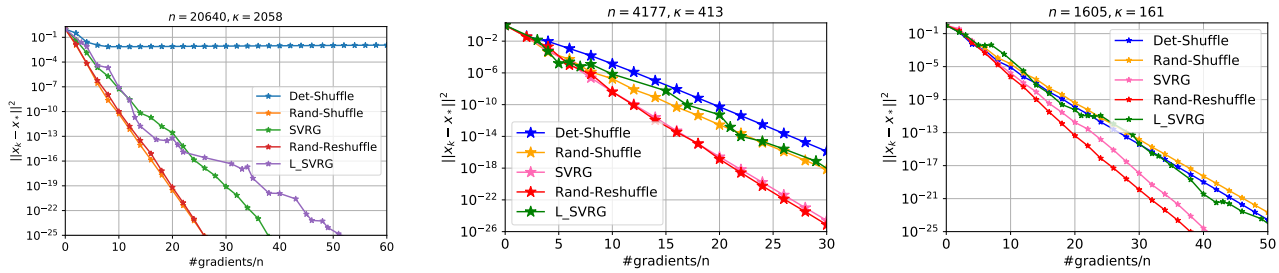
Up to a constant factor, the complexity of Det-Shuffle is the same as that of Rand-Shuffle and Rand-Reshuffle.

4 EXPERIMENTS

In our experiments we solve the regularized ridge regression problem, which has the form (1) with

$$f_i(x) = \frac{1}{2}\|A_{i,:}x - y_i\|^2 + \frac{\lambda}{2}\|x\|^2,$$

where $A \in \mathbb{R}^{n \times d}, y \in \mathbb{R}^n$ and $\lambda > 0$ is a regularization parameter. Note that this problem is strongly convex and



(a) (b) (c)

Figure 2: (a) Comparison of methods on `cadata` dataset, we set the regularization constant $\lambda = 10/n$ and carefully chosen stepsizes. (b, c) Comparison of SVRG, L-SVRG, Rand-Reshuffle, Det-Shuffle and Rand-Shuffle on `abalone` and `a1a` datasets. For each dataset we run 5 experiments and use average errors for each algorithm.

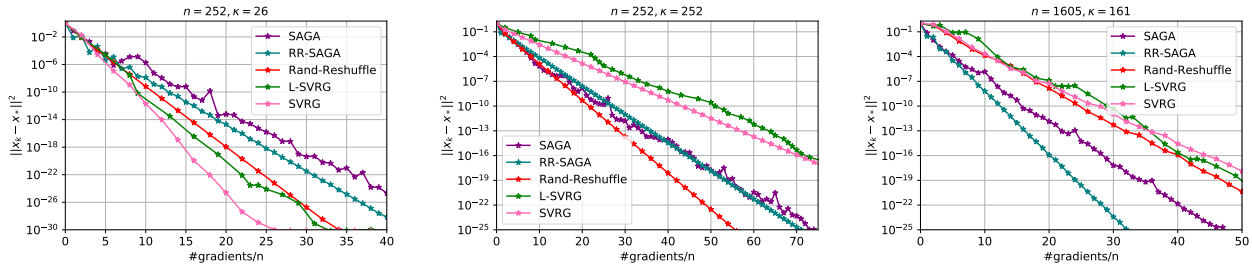


Figure 3: Comparison of SAGA, RR-SAGA, Rand-Reshuffle, L-SVRG and SVRG with optimal stepsizes on `bodyfat` dataset with different regularization constants (on the left and middle) and `a1a` (on the right).

satisfies the Assumptions 1 for $L = \max_i \|A_{i,:}\|^2 + \lambda$ and $\mu = \lambda_{\min}(A^\top A)/n + \lambda$, where $\lambda_{\min}$ is the smallest eigenvalue. To have a tighter bound on the L-smoothness constant we normalize rows of the data matrix A. We use datasets from open LIBSVM corpus [Chang and Lin, 2011]. In the plots x-axis is the number of single data gradient computation divided by n, and y-axis is the normalized error of the argument $\|x_k - x_*\|^2/\|x_0 - x_*\|^2$. In the appendix you can find the details and additional experiments.

4.1 RAND-RESHUFFLE VS RR-SAGA

In this experiment, we compare Rand-Reshuffle and RR-SAGA under an academic setting, i.e. we choose the steps that are suggested by theory. For Rand-Reshuffle we take the stepsize $\gamma = 1/\sqrt{2}Ln$ when $n \geq \frac{2L}{\mu}\frac{1}{1-\frac{\mu}{\sqrt{2}L}}$ and $\gamma = \frac{1}{2\sqrt{2}Ln}\sqrt{\frac{\mu}{L}}$ otherwise, and for RR-SAGA $\gamma = \frac{\mu}{11L^2 n}$. We can see that Rand-Reshuffle outperforms RR-SAGA in terms of the number of epochs and the number of gradient computations. Although the cost of iteration of Rand-Reshuffle is twice higher than RR-SAGA, the larger stepsize significantly impacts the total complexity. In addition, RR-SAGA needs $\mathcal{O}(nd)$ extra storage to maintain the table of gradients, which makes RR-SAGA algorithm hard to use in the big data regime.

4.2 VARIANCE REDUCED RANDOM RESHUFFLING ALGORITHMS

This section compares the variance reduced algorithms with and without random reshuffling: SAGA, RR-SAGA, SVRG, L-SVRG and Rand-Reshuffle. For each algorithm, we choose its optimal stepsizes using the grid search. To make algorithms reasonable to compare in SVRG, we set the length of the inner loop $m = n$, in L-SVRG the control update probability is $1/n$. Also, we consider only the uniform sampling version of SVRG and L-SVRG. We can see the results on Figure 3. We can see that the variance reduced algorithms perform well on this experiment, and there is no obvious leader. However, note that for SAGA and RR-SAGA, we need to have an additional $\mathcal{O}(nd)$ space to store the table of the gradients, which is a serious issue in the big data regime.

4.3 DIFFERENT VERSIONS OF SVRG

In this section, we compare different types of SVRG algorithm: SVRG, L-SVRG, Rand-Reshuffle, Rand-Shuffle and Det-Shuffle. For each algorithm we run five experiments with different random seeds with optimal stepsizes found by grid search, then we plot the best of the errors (first and third plots) and the average of the errors (second and fourth plots) on Figure **??**. We can see that Rand-Reshuffle in average outperforms other algorithms, while in some ran-

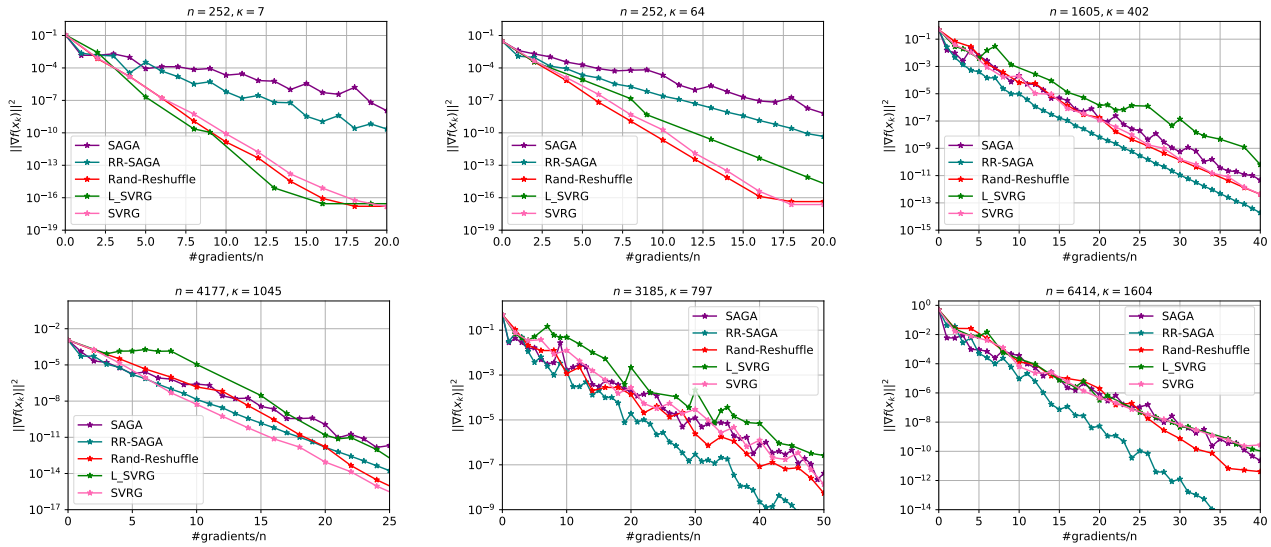


Figure 4: Comparison of SAGA, RR-SAGA, Rand-Reshuffle, L-SVRG and SVRG with optimal stepsizes on `bodyfat` dataset with different regularization constants (upper left and middle), `a1a` (upper right), `abalone` (lower left), `a3a` (lower middle) and `a5a` (lower right).

dom cases L-SVRG can perform better. Also, we can see that Rand-Shuffle is better than Det-Shuffle that coincides with theoretical findings. If the sampling in each epoch is problematic, one can shuffle data once before the training.

4.4 EXPERIMENTS WITH LOGISTIC REGRESSION

We also run experiments for the regularized logistic regression problem; i.e., for problem (1) with

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} \log\left(1 + \exp(-y_i a_i^\top x)\right) + \frac{\lambda}{2}\|x\|^2.$$

Note that the problem is L-smooth and μ-strongly convex for $L = \frac{1}{4n}\lambda_{\max}(A^\top A) + \lambda$, and $\mu = \lambda$. In these experiments (also in the ridge regression experiments) when we choose optimal stepsize, we choose the best one among $\{\frac{1}{L}, \frac{1}{2L}, \frac{1}{3L}, \frac{1}{5L}, \frac{1}{10L}\}$. For the logistic regression we do not have an explicit formula for the optimum x_* as in the ridge regression, thus in this case we compare the norm of the gradients instead. In Figure 4 we can see the performance of the variance reduced algorithms: SAGA, RR-SAGA, SVRG, L-SVRG and Rand-Reshuffle.

5 CONCLUSION

In this paper, we consider variance-reduced algorithms under random reshuffling. Our results are predominantly theoretical because these algorithms are already widely used in practice and show excellent work. We have proposed a new approach for analysis using inner product reformulation, which leads to better rates. Experimental results confirm our theoretical discoveries. Thus, we receive a deeper theoretical understanding of these algorithms' work, and we hope that this will inspire researchers to develop further and analyze these methods. The understanding of variance reduction mechanism is essential to construct accelerated versions for stochastic algorithms. We also believe that our theoretical results can be applied to other aspects of machine learning, leading to improvements in state of the art for current or future applications.

References

Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. *Neural Networks: Tricks of the Trade*, page 437–478, 2012. ISSN 1611-3349. doi: 10.1007/978-3-642-35289-8_26.

Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-dynamic programming*. Athena Scientific, 1996.

Dimitri P. Bertsekas and John N. Tsitsiklis. Gradient convergence in gradient methods with errors. *SIAM Journal on Optimization*, 10(3):627–642, January 2000. doi: 10.1137/s1052623497331063.

Léon Bottou. Curiously fast convergence of some stochastic gradient descent algorithms. 2009.

Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018. doi: 10.1137/16M1080173.

Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):1–27, 2011.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. SAGA : A fast incremental gradient method with support for non-strongly convex composite objectives. *arXiv preprint arXiv:1407.0202*, 2014a.

Aaron Defazio, Justin Domke, and Caetano. Finito: A faster, permutable incremental gradient method for big data problems. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1125–1133, Bejing, China, 22–24 Jun 2014b. PMLR.

Yoel Drori and Ohad Shamir. The complexity of finding stationary points with stochastic gradient descent. *arXiv preprint arXiv:1910.01845*, 2019.

Eduard Gorbunov, Filip Hanzely, and Peter Richtárik. A unified theory of SGD: Variance reduction, sampling, quantization and coordinate descent. In *The 23rd International Conference on Artificial Intelligence and Statistics*, 2020.

Robert M Gower, Mark Schmidt, Francis Bach, and Peter Richtárik. Variance-reduced methods for machine learning. *Proceedings of the IEEE*, 108(11):1968–1983, 2020.

Robert Mansel Gower, Nicolas Loizou, Xun Qian, Alibek Sailanbayev, Egor Shulgin, and Peter Richtárik. SGD: General analysis and improved rates. In *International Conference on Machine Learning*, pages 5200–5209. PMLR, 2019.

L. Grippo. A class of unconstrained minimization methods for neural network training. *Optimization Methods and Software*, 4(2):135–150, January 1994. doi: 10.1080/10556789408805583.

M. Gürbüzbalaban, A. Ozdaglar, and P. A. Parrilo. On the convergence rate of incremental aggregated gradient algorithms. *SIAM Journal on Optimization*, 27(2):1035–1048, Jan 2017. ISSN 1095-7189. doi: 10.1137/15m1049695.

M. Gürbüzbalaban, A. Ozdaglar, and P. A. Parrilo. Convergence rate of incremental gradient and incremental newton methods. *SIAM Journal on Optimization*, 29(4): 2542–2565, 2019. doi: 10.1137/17M1147846.

Mert Gürbüzbalaban, Asu Ozdaglar, and Pablo A Parrilo. Why random reshuffling beats stochastic gradient descent. *Mathematical Programming*, pages 1–36, 2019.

Jeff Haochen and Suvrit Sra. Random shuffling beats SGD after finite epochs. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2624–2633, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Moritz Hardt, Ben Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference on Machine Learning*, pages 1225–1234. PMLR, 2016.

Xinmeng Huang, Kun Yuan, Xianghui Mao, and Wotao Yin. An improved analysis and rates for variance reduction under without-replacement sampling orders. *Advances in Neural Information Processing Systems*, 34, 2021.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in Neural Information Processing Systems*, 26:315–323, 2013.

Xiao Li, Zhihui Zhu, Anthony Man-Cho So, and Jason D Lee. Incremental methods for weakly convex optimization. *arXiv preprint arXiv:1907.11687*, 2019.

Zhi-Quan Luo. On the convergence of the LMS algorithm with adaptive learning rate for linear feedforward networks. *Neural Computation*, 3(2):226–245, June 1991. doi: 10.1162/neco.1991.3.2.226.

O.L. Mangasarian and M.V. Solodov. Serial and parallel backpropagation convergence via nonmonotone perturbed minimization. *Optimization Methods and Software*, 4(2):103–116, 1994. doi: 10.1080/10556789408805581.

Konstantin Mishchenko, Ahmed Khaled, and Peter Richtárik. Random reshuffling: Simple analysis with vast improvements. *Advances in Neural Information Processing Systems*, 33, 2020.

Aryan Mokhtari, Mert Gürbüzbalaban, and Alejandro Ribeiro. Surpassing gradient descent provably: A cyclic incremental method with linear convergence rate. *SIAM Journal on Optimization*, 28(2):1420–1447, 2018.

Dheeraj Nagaraj, Prateek Jain, and Praneeth Netrapalli. SGD without replacement: sharper rates for general smooth convex functions. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4703–4711, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Angelia Nedić and Dimitri P. Bertsekas. Incremental subgradient methods for nondifferentiable optimization. *SIAM Journal on Optimization*, 12(1):109–138, January 2001. doi: 10.1137/s1052623499362111.

Lam M. Nguyen, Quoc Tran-Dinh, Dzung T. Phan, Phuong Ha Nguyen, and Marten van Dijk. A unified convergence analysis for shuffling-type gradient methods. *arXiv preprint arXiv:2002.08246*, 2020.

Youngsuk Park and Ernest K Ryu. Linear convergence of cyclic SAGA. *Optimization Letters*, 14(6):1583–1598, 2020.

Shashank Rajput, Anant Gupta, and Dimitris Papailiopoulos. Closing the convergence gap of SGD without replacement. *arXiv preprint arXiv:2002.10400*, 2020.

Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, page 1571–1578, Madison, WI, USA, 2012. Omnipress. ISBN 9781450312851.

Benjamin Recht and Christopher Ré. Parallel stochastic gradient algorithms for large-scale matrix completion. *Mathematical Programming Computation*, 5(2):201–226, 2013.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.

Nicolas Le Roux, Mark Schmidt, and Francis Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. *arXiv preprint arXiv:1202.6258*, 2012.

Itay Safran and Ohad Shamir. How good is SGD with random shuffling? In *Conference on Learning Theory*, pages 3250–3284. PMLR, 2020.

Ali H Sayed. Adaptive networks. *Proceedings of the IEEE*, 102(4):460–497, 2014.

Ohad Shamir. Without-replacement sampling for stochastic gradient methods. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 46–54, 2016.

Ruo-Yu Sun. Optimization for deep learning: an overview. *Journal of the Operations Research Society of China*, June 2020. doi: 10.1007/s40305-020-00309-6.

Bicheng Ying, Kun Yuan, Stefan Vlaski, and Ali H. Sayed. Stochastic learning under random reshuffling with constant step-sizes. In *IEEE Transactions on Signal Processing*, volume 67, pages 474–489, 2019.

Bicheng Ying, Kun Yuan, and Ali H Sayed. Variance-reduced stochastic learning under random reshuffling. *IEEE Transactions on Signal Processing*, 68:1390–1408, 2020.