Distributed Methods with Compressed Communication for Solving Variational Inequalities, with Theoretical Guarantees

Anonymous Author(s)
Affiliation
Address
email

Abstract

Variational inequalities in general and saddle point problems in particular are increasingly relevant in machine learning applications, including adversarial learning, GANs, transport and robust optimization. With increasing data and problem sizes necessary to train high performing models across various applications, we need to rely on parallel and distributed computing. However, in distributed training, communication among the compute nodes is a key bottleneck during training, and this problem is exacerbated for high dimensional and over-parameterized models. Due to these considerations, it is important to equip existing methods with strategies that would allow to reduce the volume of transmitted information during training while obtaining a model of comparable quality. In this paper, we present the first theoretically grounded distributed methods for solving variational inequalities and saddle point problems using compressed communication: MASHA1 and MASHA2. Our theory and methods allow for the use of both unbiased (such as Rand$k$; MASHA1) and contractive (such as Top$k$; MASHA2) compressors. New algorithms support bidirectional compressions, and also can be modified for stochastic setting with batches and for federated learning with partial participation of clients. We empirically validated our conclusions using two experimental setups: a standard bilinear min-max problem, and large-scale distributed adversarial training of transformers.

1 Introduction

1.1 The expressive power of variational inequalities

Due to their abstract mathematical nature and the associated flexibility they offer in modeling various practical problems of interests, *variational inequalities (VI)* have been an active area of research in applied mathematics for more than half a century [64, 31, 22]. It is well known that VIs can be used to formulate and study optimization problems, *saddle point problems (SPPs)*, games and fixed point problems, for example, in an elegant unifying mathematical framework [9].

Recently, a series of works by various authors [15, 26, 57, 13, 48] built a bridge between VIs/SPPs and GANs [28]. This allows to successfully transfer established insights and well-known techniques from the vast literature on VIs/SPPs, such as averaging and extrapolation, to the study of GANs. Besides their usefulness in studying GANs and alternative adversarial learning models [56], VIs/SPPs have recently attracted considerable attention of the machine learning community due to their ability to model other situations where the minimization of a single loss function does not suffice, such as auction theory [80], supervised learning with non-separable loss [39] or non-separable regularizer [7] and reinforcement learning [69, 65, 38].

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

In summary, VIs have recently become a potent tool enabling new advances in practical machine learning situations reaching beyond supervised learning where optimization problems and techniques, which can be seen as special instances of VIs and methods for solving them, reign supreme.

1.2 Training of supervised models via distributed optimization

On the other hand, in the domain of classical, and hence also much better understood, *supervised machine learning* is characterized by the fact that standard optimization techniques apply and work well, researchers and practitioners face other challenges that are currently beyond the reach of existing VI methods. Indeed, the training of modern supervised machine learning models in general, and deep neural networks in particular, is still extremely challenging. Due to their desire to improve the generalization of deployed models, machine learning engineers need to rely on training datasets of ever increasing sizes and on elaborate over-parametrized models [5]. Supporting workloads of such unprecedented magnitudes would be impossible without combining the latest advances in hardware acceleration, distributed systems and *distributed algorithm design* [83].

When training such modern supervised models in a distributed fashion, *communication cost* is often the bottleneck of the training system, and for this reason, a lot of effort was recently targeted at the design of communication efficient distributed optimization methods [44, 76, 25, 29]. A particularly successful technique for improving the communication efficiency of distributed first order optimization methods is *communication compression.* The idea behind this technique is rooted in the observation that in practical implementations it often advantageous to communicate messages compressed via (often randomized) *lossy compression techniques* instead of communicating the full messages [75, 2]. If the number of parallel workers is large enough, the noise introduced by compression is reduced, and training with compressed communication will often lead to the comparable test error while reducing the amount of communicated bits, which results in faster training, both in theory and practice [58, 29].

1.3 Two classes of compression operators

Since the paper focuses on compression methods for distributed VIs and SPPs. Let us give the main definitions. We say that a (possibly) stochastic mapping $Q : \mathbb{R}^d \to \mathbb{R}^d$ is an *unbiased compression operator* if there exists a constant $q \geq 1$ such that

$$\mathbb{E}Q(z) = z, \quad \mathbb{E}\|Q(z)\|^2 \leq q\|z\|^2, \quad \forall z \in \mathbb{R}^d. \tag{1}$$

Further, we say that a stochastic mapping $C : \mathbb{R}^d \to \mathbb{R}^d$ is a *contractive compression operator* if there exists a constant $\delta \geq 1$ such that

$$\mathbb{E}\|C(z) - z\|^2 \leq (1 - 1/\delta)\|z\|^2, \quad \forall z \in \mathbb{R}^d. \tag{2}$$

If b is the number of bits needed to represent a single float (e.g., $b = 32$ or $b = 64$), then the number of bits needed to represent a generic vector $z \in \mathbb{R}^d$ is $\|z\|_{\text{bits}} := bd$. To describe how much a compression operator reduces its input vector on average, we define the notion of expected density, denoted via $\beta^{-1} := \frac{1}{bd}\mathbb{E}\|Q(z)\|_{\text{bits}}$, where $\|Q(z)\|_{\text{bits}}$ is the number of bits needed to represent the quantized vector $Q(z)$. Note that $\beta \geq 1$. For the Rand$k$ operator [3, 10] we have $q = \beta = d/k$.

1.4 Towards communication-efficient distributed methods for VIs and SPPs

Classical VI/SPP algorithms such as the *Extra Gradient method* originally proposed by [45] and later studied by many authors [61, 40], including in a distributed environment [77, 51, 60, 73]. Among them, a number of works stand out trying to solve the communication bottleneck challenge using various approaches such as local steps, data-similarity etc.[88, 34, 16, 11, 12]. But despite the fact that the use of compression is one of the most popular communication-efficient approaches for distributed minimization problems, *no work has yet paid attention to the compression technique neither for distributed SPPs nor for VIs*, with the exception of the work [88], which relies on rounding to the nearest integer multiple of a certain quantity. This compression mechanism does not offer theoretical benefits and does not even lead to convergence to the solution due to the errors introduced through rounding persist and prevent the method from solving the problem.

2 Summary of Contributions

In this paper, we investigate whether it is possible to design communication-efficient algorithms for solving distributed VI/SPP by borrowing generic communication compression techniques (1) and (2) from the optimization literature [75, 2, 58, 29, 72] and embedding them into established, efficient methods for solving VIs/SPPs [45, 61, 40, 1]. Whether or not this is possible is an open problem. In summary,

> *we design the first provably algorithms with compression for solving general distributed VI/SPP (see Section 3, Equation 3) in the deterministic (see* (4)*), stochastic (see* (42)*) and federated (see* (52)*) regimes, supporting both unbiased (*MASHA1 *= Algorithms 1, 5, 7) and contractive (*MASHA2 *= Algorithms 2, 6, 8) compressors. Convergence of all our methods are analyzed in strongly-monotone (strongly convex - strongly concave), monotone (convex - concave) non-monotone/minty (non-convex-non-concave) cases.*

2.1 Two types of compressors

We develop two approaches for distributed VIs/SPPs depending on whether we use unbiased (1) or contractive (2) compressors, since each type of compressor demands a different algorithmic design and a different analysis. In particular, contractive compressors are notoriously hard to analyze even for optimization problems [43, 72]. Our method based on unbiased compressors is called MASHA1 (Algorithm 1), and our method based on contraction compressors is called MASHA2 (Algorithm 2).

2.2 Theoretical complexity results

We establish a number of theoretical complexity results for our methods, which we summarize in Table 1. We consider the strongly monotone (strongly convex - strongly concave), monotone (convex - concave) regimes as well as the more general non-monotone/minty (non-convex-non-concave) regime. In the strongly monotone case we obtain linear convergence results ($O(\log 1/\epsilon)$) in terms of the distance to solution, in the monotone we obtain fast sublinear convergence results ($O(1/\epsilon)$) in terms of the *gap* function, and in the non-monotone case we have sublinear convergence results ($O(1/\epsilon^2)$) in terms of the second norm of the operator. To get an estimate for the number of information transmitted, one need to multiply the estimates from Table 1 by $1/\beta$. Then we get that from the point of view of the transmitted information (and also time for communications), MASHA1 is better by a factor $\sqrt{1/\beta + 1/M}$ (M – number of workers) in comparison with the classical Extra Gradient. It means that we get an acceleration of $\min\{\sqrt{\beta}; \sqrt{M}\}$ times. For example, ADIANA from [47](the theoretical SOTA method with unbiased compressions for strongly convex minimization) has the same accelaration. The same situation is with MASHA2. The method has the same compression dependent multiplier as ECLK from [71] (the theoretical SOTA with contractive compression for minimization). Based on these facts, we hypothesize that MASHA1 and MASHA2 have unimprovable estimates (see Appendix A).

2.3 Stochastic case and variance reduction

MASHA1 and MASHA2 are designed to handle the *deterministic* setting. But often in practice, the computation of the full operators F_m is expensive, then we need to deal with *stochastic* realizations. In particular, a popular case is when each operator F_m has a finite-sum structure on its own, e.g. , finite-sum of batches. For this issue, we consider two modifications: VR-MASHA1 (Algorithm 5) and VR-MASHA2 (Algorithm 6). Both are enhanced with bespoke *variance-reduction* techniques for better theoretical and practical performance. These results can be interesting in the non-distributed case. As far as we know, we are the first who consider variance reduction for non-monotone VIs. We found only one paper on non-convex-concave saddle point problems [87] under the PL condition. See Appendix E fro details.

2.4 Federated learning and partial participation

Federated learning [44, 41] is an important and popular branch of distributed methods. Therefore, a good bonus for the algorithm is that it can be easily adapted for it. In a federated setup where the computing devices are mobile phones, tablets, personal computers etc, the importance of the communication bottleneck is even higher. Moreover, devices can not only have a bad connection, in which one needs to compress data heavily, they can simply disconnect from the learning process.

Table 1: Summary of our **iteration complexity** results for finding an ε-solution for problem (3) in the deterministic (i.e., (4)) with only device compression, deterministic with bidirectional (device-server) compression, stochastic (i.e., (4)+(42)) and federated learning/partial participation (i.e., (4)+(52)) setups. In the strongly-monotone (strongly convex - strongly convex) case, convergence is measured by the distance to the solution. In the monotone(convex-concave) case, convergence is measured in terms of the gap function (9). In non-monotone (non-convex-non-concave) case convergence is measured in terms of the norm of the operator. *Notation:* μ = constant of strong monotonicity of the operator F, L = maximum of local Lipschitz constants L_m, R = diameter (in Euclidean norm) of the optimization set, R_0 = initial distance to the solution, q = the variance parameter associated with an unbiased compressor (see (1)); δ = the variance parameter associated with a contractive compressor (see (2)); β, β = expected density (the number of times the operator compresses information); M = the number of parallel clients/nodes; r = the size of the local dataset (see (42)); b = the number of clients in Partial Participation (FL) setup. We have results with bidirectional compression also in stochastic and federated setups, but to simplify the bounds, we present bidirectional results only in the deterministic setup.

		Strongly monotone	Monotone	Non monotone
Deter. (Device) (3) + (4)	MASHA1 Alg 1 Cor 4.2	$\tilde{\mathcal{O}}\left(\beta + \sqrt{\beta + \frac{q\beta}{M}} \cdot \frac{L}{\mu}\right)$	$\mathcal{O}\left(\sqrt{\beta + \frac{q\beta}{M}} \cdot \frac{LR^2}{\varepsilon}\right)$	$\mathcal{O}\left(\left(\beta + \frac{q\beta}{M}\right) \cdot \frac{L^2R^2}{\varepsilon^2}\right)$
	MASHA2 Alg 2 Cor D.2	$\tilde{\mathcal{O}}\left(\beta + \delta\sqrt{\beta} \cdot \frac{L}{\mu}\right)$	$\mathcal{O}\left(\delta\sqrt{\beta} \cdot \frac{LR^2}{\varepsilon}\right)$	$\mathcal{O}\left(\delta^2\beta \cdot \frac{L^2R^2}{\varepsilon^2}\right)$
Deter. (Bidirect.) (3) + (4)	MASHA1 Alg 1 Cor C.3	$\tilde{\mathcal{O}}\left(\beta + \sqrt{q\beta + \frac{q^2\beta}{M}} \cdot \frac{L}{\mu}\right)$	$\mathcal{O}\left(\sqrt{q\beta + \frac{q^2\beta}{M}} \cdot \frac{LR^2}{\varepsilon}\right)$	$\mathcal{O}\left(\left(q\beta + \frac{q^2\beta}{M}\right) \cdot \frac{L^2R^2}{\varepsilon^2}\right)$
	MASHA2 Alg 2 Cor D.3	$\tilde{\mathcal{O}}\left(\beta + \delta^2\sqrt{\beta} \cdot \frac{L}{\mu}\right)$	$\mathcal{O}\left(\delta^2\sqrt{\beta} \cdot \frac{LR^2}{\varepsilon}\right)$	$\mathcal{O}\left(\delta^4\beta \cdot \frac{L^2R^2}{\varepsilon^2}\right)$
Stoch. (F-S) (3) + (4) + (42)	VR-MASHA1 Alg 5 Cor E.3	$\tilde{\mathcal{O}}\left(\beta + r + \max\{\sqrt{\beta}; \sqrt{r}\}\sqrt{1 + \frac{q}{M}} \cdot \frac{L}{\mu}\right)$	$\mathcal{O}\left(\max\{\sqrt{\beta}; \sqrt{r}\}\sqrt{1 + \frac{q}{M}} \cdot \frac{LR^2}{\varepsilon}\right)$	$\mathcal{O}\left(\left(\max\{\beta; r\}\left(1 + \frac{q}{M}\right)\right) \cdot \frac{L^2R^2}{\varepsilon^2}\right)$
	VR-MASHA2 Alg 6 Cor E.5	$\tilde{\mathcal{O}}\left(\beta + r + \max\{\sqrt{\beta}; \sqrt{r}\}\delta \cdot \frac{L}{\mu}\right)$	$\mathcal{O}\left(\max\{\sqrt{\beta}; \sqrt{r}\}\delta \cdot \frac{LR^2}{\varepsilon}\right)$	$\mathcal{O}\left(\max\{\beta; r\}\delta^2 \cdot \frac{L^2R^2}{\varepsilon^2}\right)$
FL (PP) (3) + (4)+ (52)	PP-MASHA1 Alg 7 Cor F.2	$\tilde{\mathcal{O}}\left(\frac{\beta M}{b} + \sqrt{\frac{\beta M}{b} + \frac{q\beta M}{b}} \cdot \frac{L}{\mu}\right)$	$\mathcal{O}\left(\sqrt{\frac{\beta M}{b} + \frac{q\beta M}{b}} \cdot \frac{LR^2}{\varepsilon}\right)$	$\mathcal{O}\left(\left(\frac{\beta M}{b} + \frac{q\beta M}{b}\right) \cdot \frac{L^2R^2}{\varepsilon^2}\right)$
	PP-MASHA2 Alg 8 Cor F.4	$\tilde{\mathcal{O}}\left(\frac{\beta M}{b} + \delta\sqrt{\frac{\beta M^3}{b^3}} \cdot \frac{L}{\mu}\right)$	$\mathcal{O}\left(\delta\sqrt{\frac{\beta M^3}{b^3}} \cdot \frac{LR^2}{\varepsilon}\right)$	$\mathcal{O}\left(\delta^2\beta\frac{M^3}{b^3} \cdot \frac{L^2R^2}{\varepsilon^2}\right)$

Therefore, we introduce two modifications: PP-MASHA1 (Algorithm 7) and PP-MASHA2 (Algorithm 8), that support the mode of *partial participation* of devices in the learning process. For minimization problems, a combination of quantization and partial participation occurs in [33, 68, 29]. The results are contained in Appendix F.

2.5 Bidirectional compression

Most methods, especially with contractive compressors, only use compression when transferring information from devices to the server. Meanwhile, quite often in practical situations, the transfer of information from the server to the device is also expensive [32, 81, 68]. In the literature on distributed heterogeneous minimization, we can highlighted only some works on bidirectional unbiased [68] and contractive compressors [54, 23]. Also one can note some works [90, 81] for identical/homogeneous data on workers. But most of these methods have their small drawbacks, such as only deterministic case, homogeneity not suitable for federated learning etc. All our methods MASHA1, MASHA2 and their modifications support bidirectional compression. See Appendix C and D for details.

2.6 Experiments

Toy experiments on bilinear problems show that methods with compression for minimization problems may not work (diverge) for SPPs. Also we verify that MASHA1 and MASHA2 are much better than the classical Extra Gradient with added unbiased compression. Experiments on adversarial training of large-scale transformer (ALBERT) show the practical importance of compression in distributed methods for large SPPs.

3 Problem Formulation and Assumptions

3.1 Problem formulation

We study distributed variational inequality (VI) problem

$$\text{Find } z^* \in \mathbb{R}^d \text{ such that } \langle F(z^*), z - z^* \rangle \geq 0, \quad \forall z \in \mathbb{R}^d, \tag{3}$$

where $F : \mathbb{R}^d \to \mathbb{R}^d$ is an operator with certain favorable properties (e.g., Lipschitzness and monotonicity). We assume that the training data describing F is *distributed* across M workers/nodes/clients

$$F(z) := \frac{1}{M} \sum_{m=1}^{M} F_m(z), \tag{4}$$

where $F_m : \mathbb{R}^d \to \mathbb{R}^d$ for all $m \in \{1, 2, \ldots, M\}$. For the saddle point problem

$$\min_{x \in \mathbb{R}^{d_x}} \min_{y \in \mathbb{R}^{d_y}} g(x, y) := \frac{1}{M} \sum_{m=1}^{M} g_m(x, y) \tag{5}$$

we take $z := [x, y]$ and $F_m(z) := [\nabla_x g(x, y), -\nabla_y g(x, y)]$.

3.2 Assumptions

Next, we list two key assumptions - both are standard in the literature on VIs.

Assumption 3.1 (Lipschitzness) *The operator F is L-Lipschitz continuous, i.e. for all $z_1, z_2 \in \mathbb{R}^d$ we have $\|F(z_1) - F(z_2)\| \leq L\|z_1 - z_2\|$.*

Each operator F_m is L_m-Lipschitz continuous, i.e. for all $z_1, z_2 \in \mathbb{R}^d$ it holds $\|F_m(z_1) - F_m(z_2)\| \leq L_m\|z_1 - z_2\|$. Let us define new constant $\tilde{L}$ as follows $\tilde{L}^2 = \frac{1}{M} \sum\limits_{m=1}^{M} L_m^2$.

For saddle point problems, these properties are equivalent to smoothness.

Assumption 3.2 (Monotonicity) *We need three cases of monotonicity*

(SM) Strong monotonicity. *The operator F is μ-strongly monotone, i.e. for all $z_1, z_2 \in \mathbb{R}^d$ we have $\langle F(z_1) - F(z_2), z_1 - z_2 \rangle \geq \mu\|z_1 - z_2\|^2$.*

(M) Monotonicity. *The operator F is monotone, i.e. for all $z_1, z_2 \in \mathbb{R}^d$ we have $\langle F(z_1) - F(z_2), z_1 - z_2 \rangle \geq 0$.*

(NM) Non-monotonicity. *The operator F is non-monotone (minty), if and only if there exists $z^* \in \mathbb{R}^d$ such that for all $z \in \mathbb{R}^d$ we have $\langle F(z), z - z^* \rangle \geq 0$.*

The last assumption is called the minty or variational stability condition. It is not a general non-monotonicity, but is already associated in the community with non-monotonicity [14, 37, 57, 52, 42, 36, 19], particularly with the setup, which is somewhat appropriate for GANS [50, 51, 21, 8].

4 MASHA

In this Section we present new algorithms and their convergence. Section 4.1 is devoted to the algorithm (MASHA1) with unbiased compression. Section 4.2 – to algorithm (MASHA2) with contractive compression. Appendix gives modifications for the stochastic case – Section E, and for the federated learning – Section F. Appendix A is devoted to the hypothesis about optimality of MASHA1 and MASHA2.

4.1 MASHA1: Handling Unbiased Compressors

Before presenting our algorithm, let us discuss which approaches can be used to construct it. As discussed in Sections 1 and 2, compression methods play an important role in distributed minimization problems. All these methods are modifications of the classical GD. For instance, the authors of [2] compress stochastic gradients. Therefore, it is a natural idea to use GD-type methods for VIs as well. But it is a well-known fact that GD-type methods can give bad convergence estimates (see Section B.1 from [67]) or do not converge at all (see Section 7.2 and 8.2 from [27]) even on the simplest SPPs and VIs. From a practical point of view, this approach can also fail (see QSGD and EF in Section 5.1). In the non-distributed case, this problem has long been solved and the Extra Gradient method [45, 61, 40] is used instead of GD:

$$z^{k+1/2} = z^k - \gamma F(z^k), \quad z^{k+1} = z^k - \gamma F(z^{k+1/2}). \tag{6}$$

This method is optimal for both VIs and SPPs and has an estimate of convergence $\tilde{\mathcal{O}}(L/\mu)$ in the strongly monotone case. Therefore, the second idea for the compressed method is to add compression operators to the method (6), e.g. use $Q_k(F(z^k))$ and $Q_{k+1/2}(F(z^{k+1/2}))$ instead of $F(z^k)$ and $F(z^{k+1/2})$. In Section G we analyse this method, but it gives an estimate $\tilde{\mathcal{O}}\left(1 + q/M\right) \cdot L^2/\mu^2\big)$, which is considerably worse in terms of L/μ than the original Extra Gradient method. The key problem is that in the analysis one has to deal with $\|Q_k(F(z^{k+1/2})) - Q_{k+1/2}(F(z^k))\|^2$. Without compression operators, such difference is easily evaluated using Assumption 3.1. But when the compression operators are different (in fact the same, but have different randomness) we cannot make a good estimate for this term. The idea arises to use the same randomness in both steps of the method (6), namely to substitute $Q_k(F(z^k))$ and $Q_k(F(z^{k+1/2}))$. But then $z^{k+1/2}$ depends on the randomness Q_k, and hence $Q_k(F(z^{k+1/2}))$ is biased, which further complicates the analysis. For exactly the same reasons, the various optimistic/single call modifications [70, 26, 35, 59] of the Extra Gradient method did not work for us either. We have also test the method (6) with compressions in practice (see CEG in Section 5.1), and it turns out to be worse than the method we will present below. In the end, the use of variance reduction and negative momentum techniques [1] is key in creating our algorithm. These tricks are not in themselves relevant to distributed problems, but, in our case, they help in creating MASHA1 and MASHA2.

Algorithm 1 MASHA1

Parameters: Stepsize $\gamma > 0$, parameter $\tau \in (0; 1)$, number of iterations K.
Initialization: Choose $z^0 = w^0 \in \mathcal{Z}$.
Devices send $F_m(w^0)$ to server and get $F(w^0)$
for $k = 0, 1, 2, \ldots, K-1$ **do**
  for each device m in parallel **do**
    $z^{k+1/2} = \tau z^k + (1-\tau)w^k - \gamma F(w^k)$
    Sends $g_m^k = Q_m^{\text{dev}}(F_m(z^{k+1/2}) - F_m(w^k))$ to server
  end for
  for server **do**
    Sends to devices $g^k = Q^{\text{serv}}\left[\frac{1}{M}\sum_{m=1}^M g_m^k\right]$
    Sends to devices one bit b_k : 1 with probability $1-\tau$, 0 with with probability τ
  end for
  for each device m in parallel **do**
    $z^{k+1} = z^{k+1/2} - \gamma g^k$
    If $b_k = 1$ then $w^{k+1} = z^k$, sends $F_m(w^{k+1})$ to server and gets $F(w^{k+1})$
    else $w^{k+1} = w^k$
  end for
end for

At the beginning of each iteration of MASHA1, each device knows the value of $F(w^k)$, hence it can be calculated $\bar{z}^k$ and $z^{k+1/2}$ locally. Then it sends the compressed version of the difference $F_m(z^{k+1/2}) - F_m(w^k)$ to the server, and the server does a reverse compressed broadcast. As a result, an unbiased estimate of $F(z^{k+1/2}) - F(w^k)$ appears on each node. Also, the nodes receive a bit of information b_k. This bit is generated randomly on the server and is equal to 1 with probability $1-\tau$ (where $1-\tau$ is small). Note that b_k can be generated locally, it is enough to use the same random generator and set the same seed on all devices. Then, all devices make a final update on z^{k+1}, and also update the $w^{k+1} = z^{k+1}$ point if the $b_k = 1$ or save it from the previous iteration $w^{k+1} = w^k$. In the case when the point $w^{k+1} = z^{k+1}$, we need to exchange the full values of $F_m(w^{k+1})$ in order that at the beginning of the next iteration the value of $F(w^{k+1})$ is known to all nodes. We use a possibly difference compressor on each device and also on the server. To distinguish between them, we denote the following notation: Q_m^{dev}, q_m^{dev}, β_m^{dev} and Q^{serv}, q^{serv}, q^{serv}.

Theorem 4.1 *Let Assumption 3.1 and one case of Assumption 3.2 are satisfied. Then for some step γ the following estimates on* MASHA1 *number of iterations to achieve ε-solution holds*

- *in strongly-monotone case:* $\mathbb{E}[\|z^K - z^*\|^2 + \|w^K - z^*\|^2] = \mathcal{O}([\frac{1}{1-\tau} + \frac{C_q}{\mu\sqrt{1-\tau}}]\log\frac{1}{\varepsilon})$;
- *in monotone case:* $\mathbb{E}\max_{z\in\mathcal{C}}[\langle F(u), (\frac{1}{K}\sum_{k=0}^{K-1} z^{k+1/2}) - u\rangle] = \mathcal{O}(\frac{C_q\|z^0 - z^*\|^2}{\varepsilon\sqrt{1-\tau}})$;

• *in non-monotone case:* $\mathbb{E}[\frac{1}{K}\sum_{k=0}^{K-1}\|F(w^k)\|^2] = \mathcal{O}(\frac{C_q^2\|z^0-z^*\|^2}{\varepsilon^2(1-\tau)})$;

where $C_q^2 = \frac{q^{serv}}{M^2}\sum_{m=1}^{M}(q_m^{dev}L_m^2 + (M-1)\tilde{L}^2)$.

A full listing of the algorithm, as well as a full statement of the theorem with proof, can be found in Appendix C.

The bounds in Theorem 4.1 are related to τ. Let us find optimal way to choose it. Note that (in average) once per $1/(1-\tau)$ iterations (when $b_k = 1$), we send uncompressed information. Based on this observation, we can find the best option for τ. Let us analyze the case of compressions only on the devices' side ($q^{\text{serv}} = 1$). For simplicity, we put $Q_m^{\text{dev}} = Q$ with $q_m^{\text{dev}} = q$ and $\beta_m^{\text{dev}} = \beta$, also $L_m = \tilde{L} = L$. Since compression is done only on devices, we assume that the server's broadcast is cheap and we only care about devices. Then at each iteration the device sends $\mathcal{O}\left(1/\beta + 1 - \tau\right)$ bits – each time information compressed by β times and with probability $1-\tau$ we send the full package. From where we immediately get the optimal choice for τ:

Corollary 4.2 *Let Assumption 3.1 and one case of Assumption 3.2 are satisfied. Then for some step* γ *and* $1-\tau = 1/\beta$ *the following estimates on* MASHA1 *number of iterations to achieve* ε*-solution holds*

• *in str-monotone case:* $\mathcal{O}([\beta + \sqrt{\frac{q\beta}{M} + \beta}\cdot\frac{L}{\mu}]\log\frac{1}{\varepsilon})$;

• *in monotone case:* $\mathcal{O}(\sqrt{\frac{q\beta}{M} + \beta}\cdot\frac{L\|z^0-z^*\|^2}{\varepsilon})$;

• *in non-monotone case:* $\mathcal{O}([\frac{q\beta}{M} + \beta]\frac{L^2\|z^0-z^*\|^2}{\varepsilon^2})$.

We can see that MASHA1 can outperform the uncompressed Extra Gradient method. Let us compare them in the strongly monotone case. The communication complexity of the Extra Gradient method is $\tilde{\mathcal{O}}(L/\mu)$. MASHA1 has communication complexity $\tilde{\mathcal{O}}(\sqrt{q/\beta M + 1/\beta}\cdot L/\mu)$. For practical compressors [10], $\beta \geq q$. Then once can note that MASHA1 $\mathcal{O}(\sqrt{1/M + 1/\beta}\cdot L/\mu)$ better than the uncompressed extragradient method. We think that the factor $\mathcal{O}(\sqrt{1/M + 1/\beta}\cdot L/\mu)$ is unimprovable and optimal – see Section A.

One can also consider the case of bidirectional compression ($q^{\text{serv}} \neq 1$). Table 1 (line 3) shows the result for $q^{\text{serv}} = q_m^{\text{dev}} = q$, $\beta^{\text{serv}} = \beta_m^{\text{dev}} = \beta$ and $1-\tau = 1/\beta$.

4.2 MASHA2: Handling Contractive Compressors

The use of contractive compressions is a more complex issue. In particular, it is known that if one simply put a contractive cospressor instead of an unbiased one, the method may diverge even for quadratic problems [10]. To fix this, an error compensation technique [78, 43, 79] is used. The point is to keep untransmitted information and add to new information at the next iteration. This is the key difference between MASHA2 and MASHA1. MASHA2 introduces additional sequences e^k, e_m^k for the server's and devices' error. To define contractive operators on devices and on the server, we introduce the following notation: $C_m^{\text{dev}}, \delta^{\text{dev}}, \beta^{\text{dev}}$ and $C_m^{\text{serv}}, \delta^{\text{serv}}, \beta^{\text{serv}}$.

In the case of MASHA1, an important theoretical question was how to design the algorithm to obtain convergence estimates as good as those for non-distributed methods. MASHA2 raises another problem for theoretical analysis, how to combine MASHA1 and the error feedback. The analysis of methods with error compensation techniques for minimization problem $\min_x f(x)$ is entirely tied to the existence of the function f [79, 71, 72]. In particular, the differences $f(x^k) - f(x^*)$ appear in the whole analysis and key in the technical lemmas. As a result $f(x) - f(x^*)$ is used as a convergence criterion even in the strongly convex case. But in the case of VIs there is no function f, only the operator F (the existence of $g(x,y)$ in SPP setup does not save the situation). This problem is solved in the proof of Theorem 4.3 by using an additional sequence $\|z^{k+1/2} - w^k\|$.

Theorem 4.3 *Let Assumption 3.1 and one case of Assumption 3.2 are satisfied. Then for some step* γ *the following estimates on* MASHA2 *number of iterations to achieve* ε*-solution holds*

• *strongly-monotone case:* $\mathbb{E}[\|\hat{z}^K - z^*\|^2 + \|w^K - z^*\|^2] = \mathcal{O}([\frac{1}{1-\tau} + \frac{\delta^{dev}\delta^{serv}\tilde{L}}{\mu\sqrt{1-\tau}}]\log\frac{1}{\varepsilon})$;

• *monotone case:* $\mathbb{E}\max_{z\in\mathcal{C}}[\langle F(z), \frac{1}{K}\sum_{k=0}^{K-1} z^{k+1/2}) - z\rangle] = \mathcal{O}(\frac{\delta^{dev}\delta^{serv}\tilde{L}\|z^0-z^*\|^2}{\varepsilon\sqrt{1-\tau}})$;

Algorithm 2 MASHA2

Parameters: Stepsize $\gamma > 0$, parameter τ, number of iterations K.
Initialization: Choose $z^0 = w^0 \in \mathcal{Z}$, $e_m^0 = 0$, $e^0 = 0$.
Devices send $F_m(w^0)$ to server and get $F(w^0)$
for $k = 0, 1, 2, \ldots, K-1$ **do**
  for each device m in parallel **do**
    $z^{k+1/2} = \tau z^k + (1-\tau)w^k - \gamma F(w^k)$
    Sends $g_m^k = C_m^{\text{dev}}(\gamma F_m(z^{k+1/2}) - \gamma F_m(w^k) + e_m^k)$ to server
    $e_m^{k+1} = e_m^k + \gamma F_m(z^{k+1/2}) - \gamma F_m(w^k) - g_m^k$
  end for
  for server **do**
    Sends to devices $g^k = C^{\text{serv}}\left[\frac{1}{M}\sum_{m=1}^M g_m^k + e^k\right]$
    $e^{k+1} = e^k + \frac{1}{M}\sum_{m=1}^M g_m^k - g^k$
    Sends to devices one bit b_k : 1 with probability $1-\tau$, 0 with with probability τ
  end for
  for each device m in parallel **do**
    $z^{k+1} = z^{k+1/2} - \gamma g^k$
    If $b_k = 1$ then $w^{k+1} = z^k$, sends $F_m(w^{k+1})$ to server and gets $F(w^{k+1})$
    else $w^{k+1} = w^k$
  end for
end for

- *in non-monotone case:* $\mathbb{E}[\frac{1}{K}\sum_{k=0}^{K-1}\|F(w^k)\|^2] = \mathcal{O}(\frac{(\delta^{dev}\delta^{serv})^2\tilde{L}^2\|z^0-z^*\|^2}{\varepsilon^2(1-\tau)})$.

A full listing of the algorithm, as well as a full statement of the theorem with proof, can be found in Appendix D.

The same way as in Section 4.1 we can consider only devices' and bidirectional compression. In particular, in the line 2 of Table 1 we put results for $\delta^{\text{serv}} = 1$, $\delta^{\text{dev}} = \delta$, $L_m = \tilde{L} = L$ and $1-\tau = \beta$. In the line 4 of Table 1 there are results for $\delta^{\text{serv}} = \delta^{\text{dev}} = \delta$, $L_m = \tilde{L} = L$ and $1-\tau = \beta$.

5 Experiments

5.1 Bilinear Saddle Point Problem

We start our experiments with a distributed bilinear problem, i.e. the problem (5) with

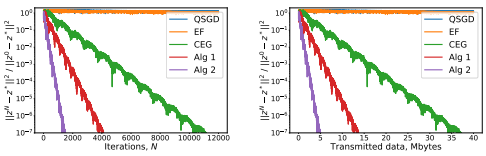
$$g_m(x,y) := x^\top A_m y + a_m^\top x + b_m^\top y + \frac{\lambda}{2}\|x\|^2 - \frac{\lambda}{2}\|y\|^2, \tag{7}$$

where $A_m \in \mathbb{R}^{d\times d}$, $a_m, b_m \in \mathbb{R}^d$. This problem is λ-strongly convex–strongly-concave and, moreover, all functions g_m are $\|A_m\|_2$-smooth. Therefore, such a distributed problem is well suited for the primary comparison of our methods. We take $d = 100$ and generate positive definite matrices A_m and vectors a_m, b_m randomly, λ is chosen as $\max_m \|A_m\|_2/10^5$.

The purpose of the experiment is to understand whether the MASHA1 and MASHA2 methods are superior to those in the literature. As a comparison, we take QGD [2] with Random 30%, classical Error Feedback [78] with Top 30% compression, as well as CEG (Section G) – Compressed Extra Gradient, each step of which we use Random 30%. In MASHA1 (Algorithm 1) we also used Random 30%, in MASHA2 (Algorithm 2) – Top 30%. See Figure 1. The stepsizes of all methods are chosen for best convergence.

We see that methods based on gradient descent (QSGD and EF) diverge. This confirms that one needs to use method specifically designed for saddle point problems (for example, the extragradient method), and not classical optimization methods. The much slower

Figure 1: Comparison MASHA1 (Algorithm 1) and MASHA2 (Algorithm 2) with Error Feedback, QGD and Compressed Extra Gradient (CEG) in iterations and in Mbytes for (7).



convergence of CEG shows the efficiency of our approach in which we compress the differences $F_m(z^{k+1/2}) - F(w^k)$. MASHA2 wins MASHA1. This shows that in practice a contractive compressor can perform better than an unbiased one with the same parameters.

5.2 Adversarial Training of Transformers

We now evaluate how compression performs for variational inequalities (and for saddle point problems, as a special case) in a more practically motivated scenario. Indeed, saddle point problems (special case of variational inequalities) have sample applications in machine learning, including *adversarial training*. We consider one of these tasks and train a *transformer-based masked language model* [82, 18, 55] using a fleet of 16 low-cost preemptible workers with T4 GPU and low-bandwidth interconnect. For this task, we use the compute-efficient adversarial training regimen proposed for transformers by [91, 53]. Formally, the adversarial formulation of the problem is the min-max problem

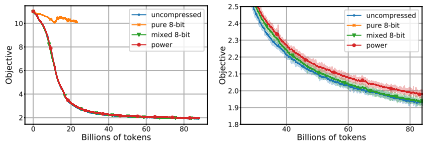
$$\min_{w} \max_{\|\rho_n\| \leq e} \frac{1}{N} \sum_{n=1}^{N} l(f(w, x_n + \rho_n, y_n)^2 + \frac{\lambda}{2}\|w\|^2,$$

where w are the weights of the model, $\{(x_n, y_n)\}_{n=1}^N$ are pairs of the training data, ρ is the so-called adversarial noise which introduces a perturbation in the data, and λ are the regularization parameters. To make our setup more realistic, we train ALBERT-large with layer sharing [46], which was recently shown to be much more communication-efficient during training [74, 20]. We train our model on a combination of Bookcorpus and Wikipedia datasets with the same optimizer (LAMB) and parameters as in the original paper [46], use the adversarial training configuration of [91], and follow system design considerations for preemptible instances [74].

In terms of communication, we consider 4 different setups for gradient compression: the "baseline" strategy with uncompressed gradients, full 8-bit quantization [17, 49], mixed 8-bit quantization, and Power compression [84] with rank r=8. For mixed 8-bit quantization and Power we only apply compression to gradient tensors with more than 2^{16} elements, sending smaller ones uncompressed. These small tensors represent layer biases and LayerNorm scales [6] that collectively amount to $\leq 1\%$ of the total gradient, but can be more difficult to compress than regular weight tensors. Finally, since Power is a biased compression algorithm, we use error feedback [43, 72] with a modified formulation proposed by [84]. For all experimental setups, we report learning curves in terms of the model training objective, similarly to [24, 74]. To quantify the differences in training loss better, we also evaluate the downstream performance for each model on several popular tasks from [85] after each model was trained on approximately 80 billion tokens. Finally, we measure the communication efficiency of each proposed strategy by measuring the average wall time per communication round when all 16 workers are active.

The learning curves in Figure 2 (upper) follow a predictable pattern, with more extreme compression techniques demonstrating slower per-iteration convergence. One curious exception to that is full 8-bit quantization, which was unable to achieve competitive training loss. The remaining three setups converge to similar loss values below 2. Both the baseline and mixed 8-bit compression show similar values in terms of downstream performance, with Power compression showing mild degradation. But in terms of information transfer time, methods using compression (especially Power) are significantly superior to the method without compression. This makes it possible to use such techniques to increase the training time without sacrificing quality.

Figure 2: **(upper left)** ALBERT training objective convergence rate with different compression algorithms; **(upper right)** ALBERT training objective convergence rate with different compression algorithms (zoomed); **(lower)** Average wall time per communication round with standard deviation over 5 repetitions and downstream evaluation scores on GLUE benchmark tasks after at 80 billion training tokens ($\approx 10^4$ optimizer steps).



Setup	Avg time	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST2	STS-B	WNLI
Baseline	8.79 ± 0.03	45.2	81.1	83.0	88.3	**89.0**	**67.8**	85.5	**89.4**	**18.3**
Full 8-bit	4.42 ± 0.07	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Mixed 8-bit	4.61 ± 0.08	**48.8**	**81.3**	**88.7**	88.1	85.2	64.3	88.3	87.5	16.9
Power	**1.57 ± 0.05**	43.9	80.5	85.6	**88.6**	86.0	47.2	**88.5**	88.5	16.9

References

[1] Ahmet Alacaoglu and Yura Malitsky. Stochastic variance reduction for variational inequality methods. *arXiv preprint arXiv:2102.08352*, 2021.

[2] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pages 1709–1720, 2017.

[3] Dan Alistarh, Torsten Hoefler, Mikael Johansson, Sarit Khirirat, Nikola Konstantinov, and Cédric Renggli. The convergence of sparsified gradient methods. In *Advances in Neural Information Processing Systems*, 2018.

[4] Zeyuan Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. *The Journal of Machine Learning Research*, 18(1):8194–8244, 2017.

[5] Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.

[6] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[7] Francis Bach, Rodolphe Jenatton, Julien Mairal, and Guillaume Obozinski. Optimization with sparsity-inducing penalties. *arXiv preprint arXiv:1108.0775*, 2011.

[8] Babak Barazandeh, Tianjian Huang, and George Michailidis. A decentralized adaptive momentum method for solving a class of min-max optimization problems. *Signal Processing*, 189:108245, 2021.

[9] Heinz H. Bauschke and Patrick L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer, second edition edition, 2017.

[10] Aleksandr Beznosikov, Samuel Horváth, Peter Richtárik, and Mher Safaryan. On biased compression for distributed learning. *arXiv preprint arXiv:2002.12410*, 2020.

[11] Aleksandr Beznosikov, Valentin Samokhin, and Alexander Gasnikov. Distributed saddle-point problems: Lower bounds, optimal algorithms and federated GANs. *arXiv preprint arXiv:2010.13112*, 2021.

[12] Aleksandr Beznosikov, Gesualdo Scutari, Alexander Rogozin, and Alexander Gasnikov. Distributed saddle-point problems under similarity. *arXiv preprint arXiv:2107.10706*, 2021.

[13] Tatjana Chavdarova, Gauthier Gidel, François Fleuret, and Simon Lacoste-Julien. Reducing noise in gan training with variance reduced extragradient. *arXiv preprint arXiv:1904.08598*, 2019.

[14] Cong D Dang and Guanghui Lan. On the convergence properties of non-Euclidean extragradient methods for variational inequalities with generalized monotone operators. *Computational Optimization and Applications*, 60(2):277–310, 2015.

[15] Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. Training GANs with optimism. In *International Conference on Learning Representations*, 2018.

[16] Yuyang Deng and Mehrdad Mahdavi. Local stochastic gradient descent ascent: Convergence analysis and communication efficiency. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1387–1395. PMLR, 2021.

[17] Tim Dettmers. 8-bit approximations for parallelism in deep learning. *ICLR*, 2015.

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[19] Jelena Diakonikolas, Constantinos Daskalakis, and Michael Jordan. Efficient methods for structured nonconvex-nonconcave min-max optimization. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 2746–2754. PMLR, 2021.

[20] Michael Diskin, Alexey Bukhtiyarov, Max Ryabinin, Lucile Saulnier, Quentin Lhoest, Anton Sinitsin, Dmitriy Popov, Dmitry Pyrkin, Maxim Kashirin, Alexander Borzunov, Albert Villanova del Moral, Denis Mazur, Ilia Kobelev, Yacine Jernite, Thomas Wolf, and Gennady Pekhimenko. Distributed deep learning in open collaborations. *CoRR*, abs/2106.10207, 2021.

[21] Zehao Dou and Yuanzhi Li. On the one-sided convergence of adam-type algorithms in non-convex non-concave min-max optimization. *arXiv preprint arXiv:2109.14213*, 2021.

[22] Francisco Facchinei and Jong-Shi Pang. *Finite-Dimensional Variational Inequalities and Complementarity Problems*. Springer Series in Operations Research. Springer, 2003.

[23] Ilyas Fatkhullin, Igor Sokolov, Eduard Gorbunov, Zhize Li, and Peter Richtárik. Ef21 with bells & whistles: Practical algorithmic extensions of modern error feedback. *arXiv preprint arXiv:2110.03294*, 2021.

[24] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*, 2021.

[25] Avishek Ghosh, Raj Kumar Maity, Arya Mazumdar, and Kannan Ramchandran. Communication efficient distributed approximate Newton method. In *IEEE International Symposium on Information Theory (ISIT)*, 2020.

[26] Gauthier Gidel, Hugo Berard, Gaëtan Vignoud, Pascal Vincent, and Simon Lacoste-Julien. A variational inequality perspective on generative adversarial networks. In *International Conference on Learning Representations*, 2019.

[27] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[28] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. In *Neural Information Processing Systems*, 2014.

[29] Eduard Gorbunov, Konstantin Burlachenko, Zhize Li, and Peter Richtárik. MARINA: Faster non-convex distributed learning with compression. In *38th International Conference on Machine Learning*, 2021.

[30] Yuze Han, Guangzeng Xie, and Zhihua Zhang. Lower complexity bounds of finite-sum optimization problems: The results and construction. *arXiv preprint arXiv:2103.08280*, 2021.

[31] P. T. Harker and J.-S. Pang. Finite-dimensional variational inequality and nonlinear complementarity problems: a survey of theory, algorithms and applications. *Mathematical programming*, 1990.

[32] Samuel Horvath, Chen-Yu Ho, Ludovit Horvath, Atal Narayan Sahu, Marco Canini, and Peter Richtárik. Natural compression for distributed deep learning. *arXiv preprint arXiv:1905.10988*, 2019.

[33] Samuel Horváth and Peter Richtárik. A better alternative to error feedback for communication-efficient distributed learning. *arXiv preprint arXiv:2006.11077*, 2020.

[34] Charlie Hou, Kiran K Thekumparampil, Giulia Fanti, and Sewoong Oh. Efficient algorithms for federated saddle point optimization. *arXiv preprint arXiv:2102.06333*, 2021.

[35] Yu-Guan Hsieh, Franck Iutzeler, Jérôme Malick, and Panayotis Mertikopoulos. On the convergence of single-call stochastic extra-gradient methods. *arXiv preprint arXiv:1908.08465*, 2019.

[36] Yu-Guan Hsieh, Franck Iutzeler, Jérôme Malick, and Panayotis Mertikopoulos. Explore aggressively, update conservatively: Stochastic extragradient methods with variable stepsize scaling. *Advances in Neural Information Processing Systems*, 33:16223–16234, 2020.

[37] Alfredo N Iusem, Alejandro Jofré, Roberto Imbuzeiro Oliveira, and Philip Thompson. Extragradient method with variance reduction for stochastic variational inequalities. *SIAM Journal on Optimization*, 27(2):686–724, 2017.

[38] Yujia Jin and Aaron Sidford. Efficiently solving MDPs with stochastic mirror descent. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119, pages 4890–4900. PMLR, 2020.

[39] Thorsten Joachims. A support vector method for multivariate performance measures. pages 377–384, 01 2005.

[40] Anatoli Juditsky, Arkadii S. Nemirovskii, and Claire Tauvel. Solving variational inequalities with stochastic mirror-prox algorithm, 2008.

[41] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[42] Aswin Kannan and Uday V Shanbhag. Optimal stochastic extragradient schemes for pseudomonotone stochastic variational inequality problems and their variants. *Computational Optimization and Applications*, 74(3):779–820, 2019.

[43] Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes signsgd and other gradient compression schemes. In *International Conference on Machine Learning*, pages 3252–3261. PMLR, 2019.

[44] Jakub Konečný, H. Brendan McMahan, Felix Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: strategies for improving communication efficiency. In *NIPS Private Multi-Party Machine Learning Workshop*, 2016.

[45] G. M. Korpelevich. The extragradient method for finding saddle points and other problems. *Matecon*, 12:747–756, 1976.

[46] Zhen-Zhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020.

[47] Zhize Li, Dmitry Kovalev, Xun Qian, and Peter Richtarik. Acceleration for compressed gradient descent in distributed and federated optimization. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5895–5904. PMLR, 13–18 Jul 2020.

[48] Tengyuan Liang and James Stokes. Interaction matters: A note on non-asymptotic local convergence of generative adversarial networks. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 907–915. PMLR, 16–18 Apr 2019.

[49] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and Bill Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *International Conference on Learning Representations*, 2018.

[50] Mingrui Liu, Youssef Mroueh, Jerret Ross, Wei Zhang, Xiaodong Cui, Payel Das, and Tianbao Yang. Towards better understanding of adaptive gradient algorithms in generative adversarial nets. *arXiv preprint arXiv:1912.11940*, 2019.

[51] Mingrui Liu, Wei Zhang, Youssef Mroueh, Xiaodong Cui, Jerret Ross, Tianbao Yang, and Payel Das. A decentralized parallel algorithm for training generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[52] Weijie Liu, Aryan Mokhtari, Asuman Ozdaglar, Sarath Pattathil, Zebang Shen, and Nenggan Zheng. A decentralized proximal point-type method for saddle point problems. *arXiv preprint arXiv:1910.14380*, 2019.

[53] Xiaodong Liu, Hao Cheng, Pengcheng He, Weizhu Chen, Yu Wang, Hoifung Poon, and Jianfeng Gao. Adversarial training for large neural language models. *arXiv preprint arXiv:2004.08994*, 2020.

[54] Xiaorui Liu, Yao Li, Jiliang Tang, and Ming Yan. A double residual compression algorithm for efficient distributed learning. In *International Conference on Artificial Intelligence and Statistics*, pages 133–143. PMLR, 2020.

[55] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019.

[56] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[57] Panayotis Mertikopoulos, Bruno Lecouat, Houssam Zenati, Chuan-Sheng Foo, Vijay Chandrasekhar, and Georgios Piliouras. Optimistic mirror descent in saddle-point problems: Going the extra(-gradient) mile. In *International Conference on Learning Representations*, 2019.

[58] Konstantin Mishchenko, Eduard Gorbunov, Martin Takáč, and Peter Richtárik. Distributed learning with compressed gradient differences. *arXiv preprint arXiv:1901.09269*, 2019.

[59] Aryan Mokhtari, Asuman E Ozdaglar, and Sarath Pattathil. Convergence rate of o(1/k) for optimistic gradient and extragradient methods in smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 30(4):3230–3251, 2020.

[60] Soham Mukherjee and Mrityunjoy Chakraborty. A decentralized algorithm for large scale min-max problems. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 2967–2972, 2020.

[61] Arkadi Nemirovski. Prox-method with rate of convergence $o(1/t)$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 15:229–251, 01 2004.

[62] Yurii Nesterov. Dual extrapolation and its applications to solving variational inequalities and related problems. *Mathematical Programming*, 109(2):319–344, 2007.

[63] Yurii Nesterov. *Lectures on convex optimization*, volume 137. Springer, 2018.

[64] J. Von Neumann and O. Morgenstern. *Theory of games and economic behavior*. Princeton University Press, 1944.

[65] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P. How, and John Vian. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70, pages 2681–2690. PMLR, 2017.

[66] Yuyuan Ouyang and Yangyang Xu. Lower complexity bounds of first-order methods for convex-concave bilinear saddle-point problems. *Mathematical Programming*, pages 1–35, 2019.

[67] Balamurugan Palaniappan and Francis Bach. Stochastic variance reduction methods for saddle-point problems. In *Advances in Neural Information Processing Systems*, pages 1416–1424, 2016.

[68] Constantin Philippenko and Aymeric Dieuleveut. Bidirectional compression in heterogeneous settings for distributed or federated learning with partial participation: tight convergence guarantees. *arXiv preprint arXiv:2006.14591*, 2020.

[69] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, 2017.

[70] Leonid Denisovich Popov. A modification of the arrow-hurwicz method for search of saddle points. *Mathematical notes of the Academy of Sciences of the USSR*, 28(5):845–848, 1980.

[71] Xun Qian, Peter Richtárik, and Tong Zhang. Error compensated distributed sgd can be accelerated. *arXiv preprint arXiv:2010.00091*, 2020.

[72] Peter Richtárik, Igor Sokolov, and Ilyas Fatkhullin. EF21: A new, simpler, theoretically better, and practically faster error feedback. *arXiv preprint arXiv:2106.05203*, 2021.

[73] Alexander Rogozin, Pavel Dvurechensky, Darina Dvinkikh, Alexander Beznosikov, Dmitry Kovalev, and Alexander Gasnikov. Decentralized distributed optimization for saddle point problems. *arXiv preprint arXiv:2102.07758*, 2021.

[74] Max Ryabinin, Eduard Gorbunov, Vsevolod Plokhotnyuk, and Gennady Pekhimenko. Moshpit sgd: Communication-efficient decentralized training on heterogeneous unreliable devices, 2021.

[75] Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[76] V. Smith, S. Forte, C. Ma, M. Takáč, M. I. Jordan, and M. Jaggi. CoCoA: A general framework for communication-efficient distributed optimization. *Journal of Machine Learning Research*, 18:1–49, 2018.

[77] Kunal Srivastava, Angelia Nedic, and Dusan Stipanovic. Distributed min-max optimization in networks. In *17th Conference on Digital Signal Processing*, 2011.

[78] Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified sgd with memory. *arXiv preprint arXiv:1809.07599*, 2018.

[79] Sebastian U Stich and Sai Praneeth Karimireddy. The error-feedback framework: Better rates for sgd with delayed gradients and compressed communication. *arXiv preprint arXiv:1909.05350*, 2019.

[80] Vasilis Syrgkanis, Alekh Agarwal, Haipeng Luo, and Robert E. Schapire. Fast convergence of regularized learning in games. In *Neural Information Processing Systems*, 2015.

[81] Hanlin Tang, Chen Yu, Xiangru Lian, Tong Zhang, and Ji Liu. Doublesqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression. In *International Conference on Machine Learning*, pages 6155–6165. PMLR, 2019.

[82] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[83] Joost Verbraeken, Matthijs Wolting, Jonathan Katzy, Jeroen Kloppenburg, Tim Verbelen, and Jan S Rellermeyer. A survey on distributed machine learning. *ACM Computing Surveys*, 2019.

[84] Thijs Vogels, Sai Praneeth Karinireddy, and Martin Jaggi. Powersgd: Practical low-rank gradient compression for distributed optimization. *Advances In Neural Information Processing Systems 32 (Nips 2019)*, 32(CONF), 2019.

[85] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[86] Blake E Woodworth and Nati Srebro. Tight complexity bounds for optimizing composite objectives. *Advances in neural information processing systems*, 29:3639–3647, 2016.

[87] Junchi Yang, Negar Kiyavash, and Niao He. Global convergence and variance-reduced optimization for a class of nonconvex-nonconcave minimax problems. *arXiv preprint arXiv:2002.09621*, 2020.

[88] Deming Yuan, Qian Ma, and Zhen Wang. Dual averaging method for solving multi-agent saddle-point problems with quantized information. *Transactions of the Institute of Measurement and Control*, 36(1):38–46, 2014.

[89] Junyu Zhang, Mingyi Hong, and Shuzhong Zhang. On lower iteration complexity bounds for the saddle point problems. *arXiv preprint arXiv:1912.07481*, 2019.

[90] Shuai Zheng, Ziyue Huang, and James Kwok. Communication-efficient distributed blockwise momentum sgd with error-feedback. *Advances in Neural Information Processing Systems*, 32:11450–11460, 2019.

[91] Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. Freelb: Enhanced adversarial training for natural language understanding. *arXiv preprint arXiv:1909.11764*, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] , mostly theoretical paper
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] , Section 3
   (b) Did you include complete proofs of all theoretical results? [Yes] , Appendix
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] , but we can only after de-anonymisation
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] , Section 5.1 and 5.2
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] , Figure 2 left
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] , but we can only after de-anonymisation
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] , Section 5.2
   (b) Did you mention the license of the assets? [N/A] , use open assets
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]