Fault Tolerant ML: Efficient Meta-Aggregation and Synchronous Training

Tehila Dahan [1] **Kfir Y. Levy** [2]

Abstract

In this paper, we investigate the challenging framework of Byzantine-robust training in distributed machine learning (ML) systems, focusing on enhancing both efficiency and practicality. As distributed ML systems become integral for complex ML tasks, ensuring resilience against Byzantine failures—where workers may contribute incorrect updates due to malice or error—gains paramount importance. Our first contribution is the introduction of the Centered Trimmed Meta Aggregator (CTMA), an efficient meta-aggregator that upgrades baseline aggregators to optimal performance levels, while requiring low computational demands. Additionally, we propose harnessing a recently developed gradient estimation technique based on a double-momentum strategy within the Byzantine context. Our paper highlights its theoretical and practical advantages for Byzantine-robust training, especially in simplifying the tuning process and reducing the reliance on numerous hyperparameters. The effectiveness of this technique is supported by theoretical insights within the stochastic convex optimization (SCO) framework and corroborated by empirical evidence.

1. Introduction

In modern machine learning (ML), the paradigm of large-scale distributed training systems has emerged as a cornerstone for advancing complex ML tasks. Distributed ML approaches enable to significantly accelerate the training process; thus facilitating the practical use of larger, more sophisticated models (Zhao et al., 2023). However, as these systems grow in scale and complexity, they become increasingly susceptible to a range of faults and errors. Moreover, distributed ML also propels collaborative learning across decentralized data sources (Bonawitz et al., 2019), which often differ in distribution, quality, and volume (Bonawitz et al., 2019). For example, data from different geographic locations, devices, or organizations can exhibit considerable variability. This poses a critical challenge: ensuring the training process is resilient to faults and errors in such distributed and heterogeneous environments. Fault-tolerant training becomes imperative to maintain the integrity, accuracy, and reliability of the learned models, especially when the stakes involve critical decision-making based on ML predictions.

The Byzantine model (Lamport et al., 2019; Guerraoui et al., 2023) provides a robust framework for devising and analyzing fault-tolerant training in distributed ML, due to its capability of capturing both random and adversarial failures. It addresses the challenge posed by Byzantine workers, who may introduce incorrect updates due to unpredictable or malicious behavior, thus risking the training process. Owing to its generality, the Byzantine setting has been widely adopted as a framework for devising and analyzing fault-tolerant ML approaches (Blanchard et al., 2017; Yin et al., 2018).

Addressing Byzantine failures in distributed machine learning (ML) requires robust aggregation rules to mitigate malicious or faulty updates from workers. Various robust aggregation rules have been developed in the past years, including Coordinate-wise Trimmed Mean (CWTM) (Yin et al., 2018), Krum (Blanchard et al., 2017), Geometric Median (GM) (Chen et al., 2017), CWMed (Yin et al., 2018), and Minimum Diameter Averaging (MDA) (Guerraoui et al., 2018). Each such aggregator brings unique qualities that may better address different types of Byzantine failures. The diversity in aggregator designs underscores the necessity of having a versatile toolbox, as no single method can efficiently handle all possible Byzantine scenarios.

The efficiency and robustness of these aggregators can be systematically characterized, as shown in Karimireddy et al. (2020; 2021); Allouah et al. (2023); Farhadkhani et al. (2022). These studies not only outline the worst-case strengths of various aggregators but also set benchmarks for optimal efficiency, providing a comprehensive analysis of existing algorithms. Furthermore, they introduce the concept of **Meta-Aggregators**—advanced mechanisms that

[*]Equal contribution [1]Department of Data and Decision Sciences, Technion, Haifa, Israel [2]Department of Electrical and Computer Engineering, Technion, Haifa, Israel. Correspondence to: Tehila Dahan <t.dahan@campus.technion.ac.il>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

enhance the robustness of base aggregators by combining or modifying their outputs. Meta-aggregators are devised to bolster the system's resilience, leveraging the strengths of underlying methods to achieve greater fault tolerance. Nevertheless, the application of existing meta-aggregators, e.g. Bucketing (Karimireddy et al., 2020) and Nearest Neighbor Mixing (NNM) (Allouah et al., 2023), comes with a high computational cost, highlighting a trade-off between enhanced robustness and efficiency. Notably, the cost of these meta-aggregators is m folds higher compared to the standard "average" aggregator, where m is the number of workers. This significant increase in computational demand hinders their practicality in high-dimensional scenarios and in situations where the number of workers is high.

The challenge of applying robust aggregators to standard stochastic gradients in synchronous robust training has been a notable concern, as highlighted in the work of Karimireddy et al. (2021). The core issue stems from Byzantine workers exploiting the variance in gradient estimations to introduce biases, potentially derailing the learning process. To counteract this, Karimireddy et al. (2021) have proposed to utilize momentum estimates rather than direct gradient estimates. This approach, underpinned by rigorous theoretical analysis, advocates for each honest machine to maintain a momentum that averages past stochastic gradients across approximately $\sqrt{T}$ iterations, where T is the total number of updates. Such a strategy achieves a significant reduction in variance by a factor of $\sqrt{T}$, effectively minimizing the "noise blanket" that Byzantine workers could exploit to inject harmful bias into the system (Karimireddy et al., 2021).

On the downside, implementing this momentum-based method within Byzantine environments unveils a critical limitation: the necessity for precise tuning of both the momentum parameter and the learning rate. This fine-tuning process requires a deep understanding of several factors, including the variance of noise, the fraction of Byzantine workers present, and the performance (or strength) of the chosen robust aggregator. Such prerequisites for effective application pose practical challenges, as they demand prior comprehensive knowledge about the system's specific characteristics and adversarial conditions. This complexity restricts the straightforward applicability of momentum-based approaches in diverse, real-world settings.

In this work, we develop new tools for Byzantine-robust training. Our focus is on enhancing efficiency and practicality. Our contributions:

- **Efficient Meta-Aggregator (CTMA):** We present "Centered Trimmed Meta Aggregator (CTMA)", a novel meta-aggregator that upgrades baseline aggregators to optimal performance levels. CTMA significantly reduces computational demands typically associated with prior meta-aggregators, aligning its efficiency with that of the conventional "average" aggregator. This development makes CTMA suitable for widespread deployment in large-scale scenarios, effectively overcoming the computational challenges of previous meta-aggregators and facilitating the implementation of robust, fault-tolerant distributed machine learning systems.
- **Advanced Gradient Estimation Technique**: Building on the insight that standard stochastic gradient methods are vulnerable to Byzantine disruptions, we suggest incorporating a recent double-momentum technique (Levy, 2023) to enhance resilience during the training process. Unlike traditional momentum-based approaches that necessitate complex parameter tuning, our approach simplifies the process by allowing the learning rate to be determined solely by the objective's smoothness parameter L. This approach not only streamlines the setup but also achieves a more substantial error reduction, further mitigating the impact of Byzantine workers and enhancing the practicality of synchronous robust training in distributed ML systems. We establish theoretical guarantees for this approach within the Stochastic-Convex-Optimization (SCO)– a fundamental framework for the design and analysis of machine learning algorithms (Shalev-Shwartz et al., 2009).

We also demonstrate the usefulness of our approach in practice.

Related Work. Addressing the Byzantine problem in distributed ML has evolved significantly, with historical information playing a pivotal role. Early work by Alistarh et al. (2018) utilizes the entire historical information to mitigate Byzantine faults effectively. Subsequently, momentum-based approaches emerged as a popular and straightforward solution to enhance robustness by leveraging a subset of $\sqrt{T}$ past gradients (Allen-Zhu et al., 2020; Farhadkhani et al., 2022; El Mhamdi et al., 2021; Karimireddy et al., 2020; 2021). Karimireddy et al. (2021) further highlights the necessity of historical information, showing that methods without it fail when addressing strongly convex objectives. This underscores the essential role of using historical data to ensure the robustness and reliability of learning algorithms in Byzantine settings.

In this context, traditional models often assume data homogeneity. However, data heterogeneity—where data across workers vary in distribution—is more common and presents significant challenges. The studies by Allouah et al. (2023) and Karimireddy et al. (2020) are particularly relevant to our work, demonstrating the effectiveness of incorporating momentum and robust aggregators in heterogeneous set-

tings. These studies also pioneered the exploration of meta-aggregators, which are specifically designed to enhance the performance of baseline robust aggregators. Karimireddy et al. (2020) introduce Bucketing, a meta-aggregator that averages each worker's output with those of its nearest neighbors, identified through a random permutation, and then sends these processed outputs to a robust aggregator. Building on this concept, Allouah et al. (2023) propose Nearest Neighbor Mixing (NNM), a meta-aggregator that averages outputs based on their nearest neighbors defined by Euclidean norms before sending them to a robust aggregator.

2. Setting

We consider a Heterogeneous distributed setting comprising m workers, where a subset $\mathcal{G}$ of them are honest. Conversely to the Homogeneous case, we assume each honest worker $i \in \mathcal{G}$ may independently draw i.i.d. samples from a distribution $\mathcal{D}^{(i)}$, and that the different data distributions may vary between workers. We consider stochastic optimization problems, focusing on smooth convex objective functions given by $f_i : \mathcal{K} \to \mathbb{R}$ for each honest worker i. Our goal is to minimize the joint objective of all honest workers by $f : \mathcal{K} \mapsto \mathbb{R}$:

$$f(\mathbf{x}) := \frac{1}{|\mathcal{G}|} \sum_{i \in \mathcal{G}} f_i(\mathbf{x}) := \frac{1}{|\mathcal{G}|} \sum_{i \in \mathcal{G}} \mathbb{E}_{\mathbf{z}^{(i)} \sim \mathcal{D}^{(i)}} f_i(\mathbf{x}; \mathbf{z}^{(i)}),$$

where $\mathcal{K} \subseteq \mathbb{R}^d$ is a compact convex set. Thus, the objective is an average of $|\mathcal{G}|$ functions $\{f_i : \mathcal{K} \mapsto \mathbb{R}\}_{i \in \mathcal{G}}$, and each such $f_i(\cdot)$ can be written as an expectation over losses $f_i(\cdot, \mathbf{z}^{(i)})$ where $\mathbf{z}^{(i)}$ is drawn from some distribution $\mathcal{D}_i$ which is unknown to the learner. For ease of notation, in what follows, we will not explicitly denote $\mathbb{E}_{\mathbf{z}^{(i)} \sim \mathcal{D}^{(i)}}[\cdot]$ but rather use $\mathbb{E}[\cdot]$ to denote the expectation w.r.t. all randomization.

We consider iterative first-order optimization algorithms that leverage the gradients of $f(\cdot)$ to generate a series of query points. The final point in this series, denoted as $\mathbf{x}_T$, serves as an estimation of the optimal solution. The performance of these algorithms is evaluated based on the expected excess loss, defined as:

$$\text{ExcessLoss} := \mathbb{E}[f(\mathbf{x}_T) - f(\mathbf{x}^*)] \ ,$$

where $\mathbf{x}^* \in \arg\min_{\mathbf{x} \in \mathcal{K}} f(\mathbf{x})$. The goal of the workers is to collaboratively ensure a small excess loss w.r.t. common objective $f(\cdot)$.

We consider a distributed environment, where each worker i has the ability to compute a stochastic gradient oracle $\mathbf{g} \in \mathbb{R}^d$, given by: $\mathbf{g}^{(i)} := \nabla f_i(\mathbf{x}; \mathbf{z}^{(i)})$, for some $\mathbf{z}^{(i)} \sim \mathcal{D}^{(i)}$, implying unbiasedness, i.e., $\mathbb{E}[\mathbf{g}^{(i)} | \mathbf{x}] = \nabla f_i(\mathbf{x})$. We focus on a centralized framework characterized by a central Parameter Server ($\mathcal{PS}$). In this setup, the $\mathcal{PS}$ communicates with all the workers in the network. Specifically, our main focus is on synchronous systems, where the $\mathcal{PS}$ waits for outputs from all workers before updating its global vector and then distributes this updated vector to the workers for their next step; in the spirit of minibatch-SGD (Dekel et al., 2012).

Lastly, we consider scenarios where up to a fraction of $\delta < 1/2$ are Byzantine workers exhibiting problematic behaviors, such as sending arbitrary or malicious information during the training process. These "Byzantine" workers may even collude to disrupt the training process. The identities of these workers are unknown. We denote $\mathcal{G} \subseteq [m]$ as the set of honest workers, with a size of $|\mathcal{G}| \geq (1 - \delta)m$. Conversely, $\mathcal{B} \subseteq [m]$ represents the set of Byzantine workers, whose size is $|\mathcal{B}| \leq \delta m$.

Notation. Throughout, $\|\cdot\|$ represents the L_2-norm, and for any natural N, we define $[N] := \{1, \ldots, N\}$. We use a compressed sum notation, where $\alpha_{1:t} := \sum_{k=1}^{t} \alpha_k$. In addition, for every $\mathbf{x} \in \mathbb{R}^d$, we denote the orthogonal projection of $\mathbf{x}$ onto a set $\mathcal{K}$ by $\Pi_{\mathcal{K}}(\mathbf{x}) := \arg\min_{\mathbf{y} \in \mathcal{K}} \|\mathbf{y} - \mathbf{x}\|^2$.

Assumptions. $\forall i \in \mathcal{G}$, we use the following assumptions:
Bounded Diameter: we assume there exists $D > 0$ such,

$$\max_{\mathbf{x}, \mathbf{y} \in \mathcal{K}} \|\mathbf{x} - \mathbf{y}\| \leq D \ . \tag{1}$$

Bounded Variance: there exists $\sigma > 0$ such that,

$$\mathbb{E}\|\nabla f_i(\mathbf{x}; \mathbf{z}^{(i)}) - \nabla f_i(\mathbf{x})\|^2 \leq \sigma^2 \ ; \quad \forall \mathbf{x} \in \mathcal{K} \ . \tag{2}$$

Heterogeneity: there exists $\xi > 0$ such that for any $\mathbf{x} \in \mathcal{K}$,

$$\frac{1}{|\mathcal{G}|} \sum_{i \in \mathcal{G}} \|\nabla f_i(\mathbf{x}) - \nabla f(\mathbf{x})\|^2 \leq \xi^2 \ . \tag{3}$$

Expectation over Smooth Functions: we assume that $f_i(\cdot)$ is an expectation of smooth functions, i.e. $\forall \mathbf{x}, \mathbf{y} \in \mathcal{K}$, $\mathbf{z}^{(i)} \in$ **Support**$\{\mathcal{D}^{(i)}\}$ there exist $L > 0$ such that,

$$\|\nabla f_i(\mathbf{x}; \mathbf{z}^{(i)}) - \nabla f_i(\mathbf{y}; \mathbf{z}^{(i)})\| \leq L\|\mathbf{x} - \mathbf{y}\| \ , \tag{4}$$

The above assumption also implies that the expected loss $f_i(\cdot)$ and the averaged loss $f(\cdot)$ are L smooth.
Bounded Smoothness Variance (Levy, 2023): note that the assumption that we make in Eq. (4) implies that, $\forall \mathbf{x}, \mathbf{y} \in \mathcal{K}$ there exists $\sigma_L^2 \in [0, L^2]$ such,

$$\mathbb{E}\left\|(\nabla f_i(\mathbf{x}; \mathbf{z}) - \nabla f_i(\mathbf{x})) - (\nabla f_i(\mathbf{y}; \mathbf{z}) - \nabla f_i(\mathbf{y}))\right\|^2 \leq \sigma_L^2 \|\mathbf{x} - \mathbf{y}\|^2 \ . \tag{5}$$

In Appendix A we show that Eq. (4) implies Eq. (5).

3. Robust Aggregation and Meta-Aggregators

In Byzantine-free distributed environments, the server typically employs gradient averaging as an aggregation rule (Dekel et al., 2012). This approach is particularly effective due to its ability to execute gradient computations in parallel and efficiently reduce the variance of noisy gradients. However, in Byzantine settings, such averaging may completely fail the training process, even in the face of a single Byzantine worker. Thus, it is crucial for the server to employ a robust aggregation rule; which leads to the following update,

$$\mathbf{w}_{t+1} = \Pi_{\mathcal{K}}\left(\mathbf{w}_t - \eta\mathcal{A}\left(\mathbf{g}_t^{(1)}, \ldots, \mathbf{g}_t^{(m)}\right)\right) .$$

Here $\mathcal{A}$ represents a robust aggregation function applied to the gradients $\mathbf{g}_t^{(1)}, \ldots, \mathbf{g}_t^{(m)}$ from the m workers at time t, and η is the learning rate.

In the context of robust methods, several works (Allouah et al., 2023; Karimireddy et al., 2021; 2020; Farhadkhani et al., 2022) have developed and characterized a range of robust aggregation rules. The key property of such aggregators is their ability to limit the variance between the output of a robust aggregator and the average output of honest workers. In our discussion, we adopt the robust definition of Karimireddy et al. (2021) with a few modifications. This refined definition, detailed in Definition 3.1, guarantees that the expected deviation of an aggregation rule from the average output of honest workers is limited to $c_\delta \geq 0$ times the variance of these workers' outputs.

Definition 3.1. (c_δ, δ)-**robust**. Assume we have m random vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathbb{R}^d$. Also assume we have an "honest" subset $\mathcal{G} \subseteq [m]$, implying $\{\mathbf{x}_i\}_{\in\, \mathcal{G}}$ are independent of each other. Finally, assume that there exists $\delta \in [0, 1/2)$ such that $|\mathcal{G}| \geq (1-\delta)m$. Moreover, assume that for any $i \in \mathcal{G}$ there exist $\rho_i \geq 0$ such that,

$$\mathbb{E}\|\mathbf{x}_i - \bar{\mathbf{x}}_{\mathcal{G}}\|^2 \leq \rho_i^2 ,$$

where $\bar{\mathbf{x}}_{\mathcal{G}} := \frac{1}{|\mathcal{G}|}\sum_{i\in\mathcal{G}} \mathbf{x}_i$. Then an aggregation rule $\mathcal{A}$ is called (c_δ, δ)-robust is for any such m random vectors it outputs $\hat{\mathbf{x}} \leftarrow \mathcal{A}(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ such that,

$$\mathbb{E}\|\hat{\mathbf{x}} - \bar{\mathbf{x}}_{\mathcal{G}}\|^2 \leq c_\delta \rho^2 ,$$

where $\rho^2 := \frac{1}{|\mathcal{G}|}\sum_{i\in\mathcal{G}} \rho_i^2$, and the expectation w.r.t. $\{\mathbf{x}_i\}_{i=1}^m$ and (possible) randomization in $\mathcal{A}$.

Note that Allouah et al. (2023) provides a different definition that encompasses a broad family of robust aggregators (see Definition 2 in Allouah et al. (2023)). In Lemma C.2, we demonstrate that any aggregator satisfying their definition will also meet our definition outlined above. Consequently, the upper bounds of c_δ for various aggregation rules are derived from the findings in Allouah et al. (2023) and detailed in Table 1. Furthermore, Allouah et al. (2023) establishes that the optimal value for c_δ is $\frac{\delta}{1-2\delta}$. As shown in Table 1, this optimal rate is achieved by the CWTM aggregator when $\delta \leq 1/3$. The performance of the other aggregators listed in Table 1 is suboptimal.

Table 1. Summary of aggregation rules (CWTM (Yin et al., 2018), Krum (Blanchard et al., 2017), GM* (ϵ-approximate GM) (Chen et al., 2017; Acharya et al., 2022), CWMed (Yin et al., 2018)) with respective c_δ values and computational costs.

AGG.	c_δ	COMPUTATIONAL COST
AVG	-	$O(dm)$
CWTM	$\frac{\delta}{1-2\delta}\left(1+\frac{\delta}{1-2\delta}\right)$	$O(dm\log m)$
KRUM	$1+\frac{\delta}{1-2\delta}$	$O(dm^2)$
GM*	$\left(1+\frac{\delta}{1-2\delta}\right)^2$	$O(dm + d\epsilon^{-2})$
CWMED	$\left(1+\frac{\delta}{1-2\delta}\right)^2$	$O(dm\log m)$

3.1. Centered Trimmed Meta Aggregator (CTMA)

To address the suboptimal performance of (c_δ, δ)-robust aggregators, previous works have proposed "meta-aggregators," which enhance the performance of baseline suboptimal aggregators. Concretely, such meta-aggregators employ a baseline (c_δ, δ)-robust aggregators in a black box manner to yield an improved robust aggregator with a factor of $O(\delta(1+c_\delta))$ instead of c_δ. Thus, even if $c_\delta = O(1)$ (and therefore suboptimal); then after meta-aggregation we receive an order optimal factor of $O(\delta)$. This, in turn, enables the crafting of a wide class of optimal aggregators, which can be highly beneficial in practice.

Currently, two approaches exist for meta-aggregation: the Nearest Neighbor Mixing (NNM) method (Allouah et al., 2023) and the Bucketing technique (Karimireddy et al., 2020). Both have proven to be useful in theory as well as in practice. Unfortunately, the computational complexity of NNM and Bucketing is quite high and requires $O(dm^2)$ computations in the worst case. This is substantially higher compared to the computational cost of most robust aggregators, as shown in Table 1. This issue severely limits the practicality of existing meta-aggregators in large-scale (large d) and massively parallel (large m) scenarios.

To mend the above issue, we propose an alternative meta-aggregator in Algorithm 1, named Centered Trimmed Meta Aggregator (CTMA). CTMA achieves performance comparable to NNM and Bucketing while requiring a computational cost of $O(dm + m\log m)$, which is similar (up to a logarithmic factor) to that of the standard average aggregator.

As outlined in Algorithm 1, the CTMA process begins by

Algorithm 1 Centered Trimmed Meta Aggregator (CTMA)

Input: Sequence of vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$, a (c_δ, δ)-robust aggregator $\mathcal{A}$, $\delta \in [0, 1/2)$.
Initialize: $\mathbf{x}_0 \leftarrow \mathcal{A}(\mathbf{x}_1, \ldots, \mathbf{x}_m)$.
Sort the sequence $\{\|\mathbf{x}_i - \mathbf{x}_0\|\}_{i=1}^m$ in non-decreasing order.
Define $S \leftarrow$ set of indices corresponding to the first $(1-\delta)m$ elements in the sorted sequence.
Compute $\hat{\mathbf{x}} = \frac{1}{|S|}\sum_{i \in S} \mathbf{x}_i$.
return $\hat{\mathbf{x}}$

sorting the sequence $\{\|\mathbf{x}_i - \mathbf{x}_0\|\}_{i=1}^m$ in non-decreasing order. It then computes the average of the first $(1-\delta)m$ vectors in this ordered sequence, selecting those based on their minimal deviation from the robust aggregator $\mathbf{x}_0$. The idea behind this approach is to redefine the role of the robust aggregator $\mathbf{x}_0$. Instead of using $\mathbf{x}_0$ directly, it serves as an anchor point representing a robust reference. By averaging its nearest neighbors, this method leverages the strength of the collective while preserving robustness, leading to enhanced performance.

It is immediate to see that Algorithm 1 requires $O(dm + m \log m)$ computations, in addition to computing $\mathbf{x}_0$. Furthermore, as a post-processing approach, CTMA can be effectively combined with either NNM or Bucketing without any additional cost, offering a versatile solution in optimizing robust aggregation processes. Note that the CTMA can be applied to a wide range of optimization problems, including non-convex problems.

Lemma 3.2. *Under the assumptions outlined in Definition 3.1, if CTMA receives a (c_δ, δ)-robust aggregator, $\mathcal{A}$; then the output of CTMA, $\hat{\mathbf{x}}$, is $(16\delta(1+c_\delta), \delta)$-robust.*

Proof. For simplicity lets assume w.l.o.g. that $|\mathcal{G}| = (1-\delta)m = |S|$. At Remark 3.3, we describe how to extend to the general case where $|\mathcal{G}| \geq (1-\delta)m = |S|$.

We denote $\mathbf{y}_i := \mathbf{x}_i - \mathbf{x}_0$.

$$\begin{aligned}
\hat{\mathbf{x}} - \bar{\mathbf{x}}_{\mathcal{G}} &= \frac{1}{|S|}\sum_{i \in S} \mathbf{x}_i - \bar{\mathbf{x}}_{\mathcal{G}} \\
&= \mathbf{x}_0 - \bar{\mathbf{x}}_{\mathcal{G}} + \frac{1}{|\mathcal{G}|}\sum_{i \in S}(\mathbf{x}_i - \mathbf{x}_0) \quad (|S| = |\mathcal{G}|) \\
&= -\frac{1}{|\mathcal{G}|}\sum_{i \in \mathcal{G}} \mathbf{y}_i + \frac{1}{|\mathcal{G}|}\sum_{i \in S} \mathbf{y}_i \\
&= -\frac{1}{|\mathcal{G}|}\sum_{i \in \mathcal{G}} \mathbf{y}_i + \frac{1}{|\mathcal{G}|}\sum_{i \in \mathcal{G}} \mathbf{y}_i - \frac{1}{|\mathcal{G}|}\sum_{i \in \mathcal{G}\setminus S} \mathbf{y}_i + \frac{1}{|\mathcal{G}|}\sum_{i \in S\setminus \mathcal{G}} \mathbf{y}_i \\
&= -\frac{1}{|\mathcal{G}|}\sum_{i \in \mathcal{G}\setminus S} (\mathbf{x}_i - \mathbf{x}_0) + \frac{1}{|\mathcal{G}|}\sum_{i \in S\setminus \mathcal{G}} (\mathbf{x}_i - \mathbf{x}_0).
\end{aligned}$$

Taking the squared norm of both sides and applying the Jensen's inequality, we obtain:

$$\begin{aligned}
\|\hat{\mathbf{x}} - \bar{\mathbf{x}}_{\mathcal{G}}\|^2 &= \left\| -\frac{1}{|\mathcal{G}|}\sum_{i \in \mathcal{G}\setminus S} (\mathbf{x}_i - \mathbf{x}_0) + \frac{1}{|\mathcal{G}|}\sum_{i \in S\setminus \mathcal{G}} (\mathbf{x}_i - \mathbf{x}_0) \right\|^2 \\
&\leq \frac{2|S\setminus\mathcal{G}|}{|\mathcal{G}|^2}\sum_{i \in S\setminus\mathcal{G}} \|\mathbf{x}_i - \mathbf{x}_0\|^2 + \frac{2|\mathcal{G}\setminus S|}{|\mathcal{G}|^2}\sum_{i \in \mathcal{G}\setminus S} \|\mathbf{x}_i - \mathbf{x}_0\|^2 .
\end{aligned}$$

Note that $|\mathcal{G}\setminus S| = |\mathcal{G} \cup S| - |S| \leq m - |\mathcal{G}| = |\mathcal{B}|$ and in a similar way $|S\setminus\mathcal{G}| \leq |\mathcal{B}|$. Therefore,

$$\|\hat{\mathbf{x}} - \bar{\mathbf{x}}_{\mathcal{G}}\|^2 \leq \frac{4\delta}{|\mathcal{G}|}\sum_{i \in S\setminus\mathcal{G}} \|\mathbf{x}_i - \mathbf{x}_0\|^2 + \frac{4\delta}{|\mathcal{G}|}\sum_{i \in \mathcal{G}\setminus S} \|\mathbf{x}_i - \mathbf{x}_0\|^2 .$$

Next, we show that there exists an injective function $\Phi : S\setminus\mathcal{G} \to \mathcal{G} \cap \bar{S}$. To achieve this, we first denote $|\mathcal{B} \cap S| = q$. It is important to note that $\mathcal{B} \cap S = S \setminus \mathcal{G}$, which leads us to derive the bound for $|\mathcal{G} \cap \bar{S}|$:

$$\begin{aligned}
|\bar{S} \cap \mathcal{G}| &= |\bar{S}| - |\bar{S} \cap \mathcal{B}| = |\mathcal{B}| - |\mathcal{B} \setminus (S \cap \mathcal{B})| \\
&= |\mathcal{B}| - (|\mathcal{B}| - |S \cap \mathcal{B}|) = q .
\end{aligned}$$

Consequently, it implies that $|\mathcal{G} \cap \bar{S}| = |S \setminus \mathcal{G}| = q$. Therefore, we can assert the existence of an injective function $\Phi : S \setminus \mathcal{G} \to \mathcal{G} \cap \bar{S}$. Further, by the definition of the set S, $\forall i \in \mathcal{G} \cap \bar{S}, \forall j \in S\setminus\mathcal{G}$ we have that $\|\mathbf{x}_i - \mathbf{x}_0\| \geq \|\mathbf{x}_j - \mathbf{x}_0\|$. Thus,

$$\begin{aligned}
\|\hat{\mathbf{x}} - \bar{\mathbf{x}}_{\mathcal{G}}\|^2 &\leq \frac{4\delta}{|\mathcal{G}|}\Big(\sum_{i \in S\setminus\mathcal{G}} \|\mathbf{x}_{\Phi(i)} - \mathbf{x}_0\|^2 + \sum_{i \in \mathcal{G}\setminus S} \|\mathbf{x}_i - \mathbf{x}_0\|^2\Big) \\
&\leq \frac{8\delta}{|\mathcal{G}|}\sum_{i \in \mathcal{G}} \|\mathbf{x}_i - \mathbf{x}_0\|^2 .
\end{aligned}$$

Taking the expectations of both sides gives us the following:

$$\begin{aligned}
\mathbb{E}\|\hat{\mathbf{x}} - \bar{\mathbf{x}}_{\mathcal{G}}\|^2 &\leq \frac{8\delta}{|\mathcal{G}|}\sum_{i \in \mathcal{G}} \mathbb{E}\|\mathbf{x}_i - \mathbf{x}_0\|^2 \\
&\leq \frac{16\delta}{|\mathcal{G}|}\sum_{i \in \mathcal{G}} \mathbb{E}\|\mathbf{x}_i - \bar{\mathbf{x}}_{\mathcal{G}}\|^2 + \frac{16\delta}{|\mathcal{G}|}\sum_{i \in \mathcal{G}} \mathbb{E}\|\bar{\mathbf{x}}_{\mathcal{G}} - \mathbf{x}_0\|^2 \\
&\leq 16\delta\rho^2 + 16\delta c_\delta \rho^2 = 16\delta(1+c_\delta)\rho^2 ,
\end{aligned}$$

where the second inequality uses $\|\mathbf{a} + \mathbf{b}\|^2 \leq 2\|\mathbf{a}\|^2 + 2\|\mathbf{b}\|^2$, which holds $\forall \mathbf{a}, \mathbf{b} \in \mathbb{R}^d$. The last inequality stems from the assumption in Def. 3.1.

Remark 3.3. CTMA can also receive an upper bound estimate of δ and adjust the size of S to $(1-\delta)m$. In our analysis, we assume, without loss of generality, that this estimate matches the true size of $\mathcal{B}$. Should this not be the case, our proof can alternatively rely on a subset of $\mathcal{G}$ of size $(1-\delta)m$ without affecting the validity of our conclusions.

□

4. Synchronous Robust Training

Previous works have shown that applying a robust-aggregator to the standard stochastic gradients is bound to fail (Karimireddy et al., 2020). This is due to the fact that Byzantine workers may make use of the "noise blanket" of standard gradient estimators in order to inject harmful bias, which may totally fail the learning process. Nevertheless, Karimireddy et al. (2020); Allouah et al. (2023) have devised an elegant and simple solution to this issue in the form of aggregating momentum estimates rather than gradient estimates. Concretely, the theoretical analysis in this work suggests that each (honest) worker maintains a momentum that effectively averages the past $\approx \sqrt{T}$ stochastic gradients, which leads to a variance reduction by a factor of $\sqrt{T}$. This shrinking of the "noise blanket" limits the harmful bias that may be injected by Byzantine workers. Here we take a similar approach. Nevertheless, instead of using the momentum, we make use of a recent approach called μ^2-SGD (Levy, 2023), which enables a more aggressive error reduction of factor t compared to standard gradient estimates. One benefit of this approach over utilizing momentum, is that it will allow us to pick the momentum parameter in a completely parameter-free manner, and pick the learning rate only based on the smoothness L; Conversely, previous methods that rely on momentum require picking both the momentum and the learning rate based on L, on c_δ and on the variance of stochastic gradients (Karimireddy et al., 2020; Allouah et al., 2023). Next, we overview this approach and describe and analyze it in the context of Synchronous Robust training.

4.1. μ^2-SGD

The μ^2-SGD is a variant of standard SGD with several modifications. Its update rule is of the following form, $\mathbf{w}_1 = \mathbf{x}_1 \in \mathcal{K}$, and $\forall t > 1$,

$$\mathbf{w}_{t+1} = \Pi_{\mathcal{K}}\left(\mathbf{w}_t - \eta\alpha_t\mathbf{d}_t\right), \ \mathbf{x}_{t+1} = \frac{1}{\alpha_{1:t+1}}\sum_{k\in[t+1]}\alpha_k\mathbf{w}_k \ .$$

Here, $\{\alpha_t > 0\}_t$ are importance weights that may unequally emphasize different update steps; concretely we will employ $\alpha_t \propto t$, which puts more emphasis on the more recent updates. Moreover, the $\{\mathbf{x}_t\}_t$'s are a sequence of weighted averages of the iterates $\{\mathbf{w}_t\}_t$, and $\mathbf{d}_t$ is an estimate for the gradient at the average point, i.e. of $\nabla f(\mathbf{x}_t)$. This is different than standard SGD, which employs estimates for the gradients at the iterates, i.e. of $\nabla f(\mathbf{w}_t)$. This approach is related to a technique called Anytime-GD (Cutkosky, 2019), which is strongly-connected to the notions of momentum and acceleration (Cutkosky, 2019; Kavis et al., 2019).

While in the natural stochastic version of Anytime-GD, one would use the estimate $\nabla f(\mathbf{x}_t; \mathbf{z}_t)$; the μ^2-SGD approach suggests to employ a variance reduction mechanism to yield a *corrected momentum* estimate $\mathbf{d}_t$. This is done as follows:

Algorithm 2 Synchronous Robust μ^2-SGD

Input: Learning rate $\eta_t > 0$, starting point $\mathbf{x}_1 \in \mathcal{K}$, number of steps T, importance weights $\{\alpha_t\}_t$, corrected momentum weights $\{\beta_t\}_t$, (c_δ, δ)-robust aggregation function $\mathcal{A}$.
Initialize: Set $\mathbf{w}_1 = \mathbf{x}_1$, draw $\mathbf{z}_1^{(i)} \sim \mathcal{D}$, set $\mathbf{d}_1^{(i)} = \nabla f(\mathbf{x}_1; \mathbf{z}_1^{(i)})$, $\forall i \in [m]$.
for $t = 1$ **to** T **do**
  ▷ *server update:*
  $\mathbf{w}_{t+1} = \Pi_{\mathcal{K}}(\mathbf{w}_t - \eta_t\alpha_t\mathcal{A}(\mathbf{d}_t^{(1)}, \ldots, \mathbf{d}_t^{(m)}))$
  $\mathbf{x}_{t+1} = \frac{1}{\alpha_{1:t+1}}\sum_{k=1}^{t+1}\alpha_k\mathbf{w}_k$
  for $i = 1$ **to** m **do**
    ▷ *worker update:*
    if $i \in \mathcal{G}$ **then**
      Draw $\mathbf{z}_{t+1}^{(i)} \sim \mathcal{D}$
      Compute $\mathbf{g}_{t+1}^{(i)} = \nabla f_i(\mathbf{x}_{t+1}; \mathbf{z}_{t+1}^{(i)})$
      Compute $\tilde{\mathbf{g}}_t^{(i)} = \nabla f_i(\mathbf{x}_t; \mathbf{z}_{t+1}^{(i)})$
      Update $\mathbf{d}_{t+1}^{(i)} = \mathbf{g}_{t+1}^{(i)} + (1-\beta_{t+1})(\mathbf{d}_t^{(i)} - \tilde{\mathbf{g}}_t^{(i)})$
    else
      Return *
    end if
  end for
end for
Output: $\mathbf{x}_T$

$\mathbf{d}_1 := \nabla f(\mathbf{x}_1; \mathbf{z}_1)$, and $\forall t > 2$,

$$\mathbf{d}_t = \nabla f(\mathbf{x}_t; \mathbf{z}_t) + (1-\beta_t)(\mathbf{d}_{t-1} - \nabla f(\mathbf{x}_{t-1}; \mathbf{z}_t)) \ ,$$

where $\beta_t \in [0, 1]$ are called *corrected momentum* weights. It can be shown by induction that $\mathbb{E}[\mathbf{d}_t] = \mathbb{E}[\nabla f(x_t)]$; however, in general, $\mathbb{E}[\mathbf{d}_t \mid x_t] \neq \nabla f(x_t)$ (in contrast to standard SGD estimators). Nevertheless, Levy (2023) demonstrated that by choosing *corrected momentum* weights $\beta_t := 1/t$, the above estimate enjoys an aggressive error reduction. Specifically, $\mathbb{E}\|\varepsilon_t\|^2 := \mathbb{E}\|\mathbf{d}_t - \nabla f(\mathbf{x}_t)\|^2 \leq O(\tilde{\sigma}^2/t)$ at step t, where $\tilde{\sigma}^2 \leq O(\sigma^2 + D^2\sigma_L^2)$. Implying that the variance decreases with t, contrasting with standard SGD where the variance $\mathbb{E}\|\varepsilon_t^{\text{SGD}}\|^2 := \mathbb{E}\|\mathbf{g}_t - \nabla f(\mathbf{x}_t)\|^2$ remains uniformly bounded.

4.2. Synchronous Robust μ^2-SGD

We consider a synchronous training approach in the spirit of Minibatch-SGD (Dekel et al., 2012), where the $\mathcal{PS}$ updates its global vector by aggregating information from all workers. This aggregation occurs only after receiving outputs from all workers. Afterward, the $\mathcal{PS}$ sends the updated global vector back to each worker, allowing them to compute their next output.

Algorithm 2 describes the adapted μ^2-SGD algorithm for the synchronous Byzantine setting. In this approach, each

worker computes its own corrected momentum $\mathbf{d}_t^{(i)}$ independently using its own data samples:

$$\mathbf{d}_t^{(i)} = \nabla f_i(\mathbf{x}_t; \mathbf{z}_t^{(i)}) + (1-\beta_t)(\mathbf{d}_{t-1}^{(i)} - \nabla f_i(\mathbf{x}_{t-1}; \mathbf{z}_t^{(i)}))\ .$$

Once the outputs from all workers are received, the $\mathcal{PS}$ then proceeds to update the new query point $\mathbf{x}_{t+1}$:

$$\mathbf{w}_{t+1} = \Pi_{\mathcal{K}}\left(\mathbf{w}_t - \eta\alpha_t \mathcal{A}\left(\mathbf{d}_t^{(1)} \dots, \mathbf{d}_t^{(m)}\right)\right),$$
$$\mathbf{x}_{t+1} = \frac{1}{\alpha_{1:t+1}} \sum_{k\in[t+1]} \alpha_k \mathbf{w}_k\ ,$$

where we are assumed that $\mathcal{A}\left(\mathbf{d}_t^{(1)}, \dots, \mathbf{d}_t^{(m)}\right)$ is a (c_δ, δ)-robust aggregator (Def. 3.1).

The motivation for utilizing the μ^2-SGD algorithm in the Byzantine setting lies in its dual mechanism, which incorporates weighted averaging over the entire history of both query points and gradients. This allows the μ^2-SGD algorithm to perform careful gradient steps with small changes, thereby enhancing the stability of the gradient descent process. This stability leads to aggressive stochastic variance reduction, which can be valuable for identifying Byzantine noise (or malicious workers) over time.

Theorem 4.1. *For each worker $i \in \mathcal{G}$, assume a convex function $f_i : \mathcal{K} \mapsto \mathbb{R}$ on a convex set $\mathcal{K}$ with bounded diameter D. Under the assumptions given in Equations (2), (4), and (5), the implementation of Algorithm 2 with parameters $\{\alpha_t = t\}_t$ and $\{\beta_t = 1/t\}_t$ guarantees the following for every $t \in [T]$ and each honest worker $i \in \mathcal{G}$:*

$$\mathbb{E}\left\|\varepsilon_t^{(i)}\right\|^2 = \mathbb{E}\left\|\mathbf{d}_t^{(i)} - \nabla f_i(\mathbf{x}_t)\right\|^2 \le \tilde{\sigma}^2/t\ ,$$
$$\mathbb{E}\left\|\frac{1}{|\mathcal{G}|}\sum_{i\in\mathcal{G}} \varepsilon_t^{(i)}\right\|^2 \le \tilde{\sigma}^2/t|\mathcal{G}|\ ,$$

where $\varepsilon_t^{(i)} = \mathbf{d}_t^{(i)} - \nabla f_i(\mathbf{x}_t^{(i)})$ and $\tilde{\sigma}^2 := 2\sigma^2 + 8D^2\sigma_L^2$.

Theorem 4.1 shows that this method is consistent with the guarantees of the non-distributed μ^2-SGD (Levy, 2023), preserving the same stochastic error $\mathbb{E}\left\|\varepsilon_t^{(i)}\right\| := \mathbb{E}\left\|\mathbf{d}_t^{(i)} - \nabla f_i(\mathbf{x}_t)\right\|$ for each honest worker. Furthermore, the collective error across honest workers results in a variance reduction proportional to the number of workers, aligning with the principles of mini-batch SGD (Dekel et al., 2012).

As previously discussed, the double mechanism of the μ^2-SGD method offers significant variance reduction over the standard mini-batch SGD by minimizing stochastic variance at each step t. Consequently, integrating the μ^2-SGD algorithm with a (c_δ, δ)-robust aggregator ensures a consistent reduction in variance between the filter's output and the actual gradient, as demonstrated in Lemma 4.2.

Lemma 4.2. *Under the Byzantine assumption where $\delta < 1/2$, let $\mathcal{A}$ be a (c_δ, δ)-robust aggregation rule, and let $f : \mathcal{K} \mapsto \mathbb{R}$ be a convex function, where $\mathcal{K}$ is a convex set with bounded diameter D. Presuming that the assumptions in Equations (2), (3), (4), and (5) hold, invoking Algorithm 2 with $\{\alpha_t = t\}_t$ and $\{\beta_t = 1/t\}_t$ ensures for any $t \in [T]$:*

$$\mathbb{E}\left\|\hat{\mathbf{d}}_t - \nabla f(\mathbf{x}_t)\right\|^2 \le \frac{4\tilde{\sigma}^2}{tm} + \frac{12c_\delta\tilde{\sigma}^2}{t} + 6c_\delta\xi^2\ ,$$

where $\hat{\mathbf{d}}_t = \mathcal{A}\left(\mathbf{d}_t^{(1)}, \dots, \mathbf{d}_t^{(m)}\right)$ and $\tilde{\sigma}^2 := 2\sigma^2 + 8D^2\sigma_L^2$.

Proof. We start by determining the appropriate value for ρ as outlined in Definition 3.1, where $\bar{\mathbf{d}}_{\mathcal{G}} := \frac{1}{|\mathcal{G}|}\sum_{i\in\mathcal{G}} \mathbf{d}_t^{(i)}$:

$$\begin{aligned}
&\mathbb{E}\left\|\mathbf{d}_t^{(i)} - \bar{\mathbf{d}}_{\mathcal{G}}\right\|^2 \\
&\le 3\mathbb{E}\left\|\mathbf{d}_t^{(i)} - \nabla f_i(\mathbf{x}_t)\right\|^2 + 3\mathbb{E}\left\|\nabla f(\mathbf{x}_t) - \bar{\mathbf{d}}_{\mathcal{G}}\right\|^2 \\
&+ 3\mathbb{E}\left\|\nabla f_i(\mathbf{x}_t) - \nabla f(\mathbf{x}_t)\right\|^2 \\
&\le \frac{3\tilde{\sigma}^2}{t} + \frac{3\tilde{\sigma}^2}{t|\mathcal{G}|} + 3\mathbb{E}\left\|\nabla f_i(\mathbf{x}_t) - \nabla f(\mathbf{x}_t)\right\|^2 \\
&\le \frac{6\tilde{\sigma}^2}{t} + 3\mathbb{E}\left\|\nabla f_i(\mathbf{x}_t) - \nabla f(\mathbf{x}_t)\right\|^2\ .
\end{aligned}$$

The first and inequality uses $\|\mathbf{a}+\mathbf{b}+\mathbf{c}\|^2 \le 3\|\mathbf{a}\|^2 + 3\|\mathbf{b}\|^2 + 3\|\mathbf{c}\|^2$, which holds $\forall \mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^d$. The second inequality follows Theorem 4.1 and the assumption in Eq. (3). The third inequality is due to the fact that $|\mathcal{G}| \ge 1$. Thus,

$$\begin{aligned}
&\frac{1}{|\mathcal{G}|}\sum_{i\in\mathcal{G}} \mathbb{E}\left\|\mathbf{d}_t^{(i)} - \bar{\mathbf{d}}_{\mathcal{G}}\right\|^2 \\
&\le \frac{6\tilde{\sigma}^2}{t} + \frac{3}{|\mathcal{G}|}\sum_{i\in\mathcal{G}} \mathbb{E}\left\|\nabla f_i(\mathbf{x}_t) - \nabla f(\mathbf{x}_t)\right\|^2 \\
&\le \frac{6\tilde{\sigma}^2}{t} + 3\xi^2\ .
\end{aligned}$$

Accordingly, we set $\rho^2 := \frac{6\tilde{\sigma}^2}{t} + 3\xi^2$. Following Definition 3.1, we have,

$$\begin{aligned}
&\mathbb{E}\left\|\hat{\mathbf{d}}_t - \nabla f(\mathbf{x}_t)\right\|^2 \le 2\mathbb{E}\left\|\hat{\mathbf{d}}_t - \bar{\mathbf{d}}_{\mathcal{G}}\right\|^2 + 2\mathbb{E}\left\|\bar{\mathbf{d}}_{\mathcal{G}} - \nabla f(\mathbf{x}_t)\right\|^2 \\
&\le 2c_\delta\left(\frac{6\tilde{\sigma}^2}{t} + 3\xi^2\right) + 2\mathbb{E}\left\|\frac{1}{|\mathcal{G}|}\sum_{i\in\mathcal{G}} \varepsilon_t^{(i)}\right\|^2 \\
&\le \frac{12c_\delta\tilde{\sigma}^2}{t} + \frac{2\tilde{\sigma}^2}{t|\mathcal{G}|} + 6c_\delta\xi^2 \\
&\le \frac{12c_\delta\tilde{\sigma}^2}{t} + \frac{4\tilde{\sigma}^2}{tm} + 6c_\delta\xi^2\ ,
\end{aligned}$$

where the first inequality uses $\|\mathbf{a}+\mathbf{b}\|^2 \le 2\|\mathbf{a}\|^2 + 2\|\mathbf{b}\|^2$, which holds $\forall \mathbf{a}, \mathbf{b} \in \mathbb{R}^d$. The third follows Theorem 4.1. The last inequality utilizes the fact that $|\mathcal{G}| \ge (1-\delta)m \ge m/2$ since $\delta < 1/2$. □

This lemma breaks down the error between the filter's output and the actual gradient into three components: the collective stochastic error across all honest workers, c_δ times the stochastic error for an individual honest worker, and c_δ times the heterogeneity variance. In the absence of Byzantine workers ($\delta = 0$), c_δ can be reduced to zero if $c_\delta \leq O(\delta)$. This condition can be met by using an appropriate robust aggregator (see Table 1) or by applying a meta-aggregator like CTMA with a suitable robust aggregator. This aligns with the ideal μ^2-SGD analysis in a Byzantine-free setting (Levy, 2023).

Theorem 4.3 (Synchronous Byzantine μ^2-SGD)**.** *Assume f is convex. Also, let us make the same assumptions as in Thm. 4.1 and Lemma 4.2, and let us denote $G^* := \|\nabla f(\mathbf{x}^*)\|$, where $\mathbf{x}^* \in \arg\min_{\mathbf{x}\in\mathcal{K}} f(\mathbf{x})$. Then invoking Algorithm 2 with $\{\alpha_t = t\}_t$ and $\{\beta_t = 1/t\}_t$, and using a learning rate $\eta \leq \frac{1}{4LT}$ guarantees,*

$$\mathbb{E}[\Delta_T] \leq O\left(\frac{G^* D + LD^2}{T} + \frac{D\tilde{\sigma}}{\sqrt{T}}\sqrt{c_\delta + \frac{1}{m}} + \sqrt{c_\delta} D\xi\right)$$

where $\Delta_T := f(\mathbf{x}_T) - f(\mathbf{x}^*)$ and $\tilde{\sigma}^2 := 2\sigma^2 + 8D^2\sigma_L^2$.

Remark 4.4. Theorem 4.3 provides optimal excess loss bound when $c_\delta \leq O(\delta)$ (Karimireddy et al., 2021). Additionally, this bound is consistent with the optimal analysis in a Byzantine-free setting ($\delta = 0$) (Levy, 2023; Dekel et al., 2012).

Remark 4.5. Theorem 4.3 indicates that the learning rate solely relies on the smoothness parameter L. This is in contrast to the standard momentum approach (e.g., Karimireddy et al. (2020); Allouah et al. (2023)), which necessitates prior knowledge of L, σ, and c_δ to set the learning rate and momentum parameter.

Remark 4.6. In Appendix B.2, we demonstrate that our approach can derive a wide range of learning rates $\eta \in [\eta_{min}, \eta_{max}]$ that ensure optimal convergence. Unlike the standard momentum method, which requires a low-range of learning rates such that $\eta_{max}/\eta_{min} = O(1)$ (Karimireddy et al., 2020) to achieve an order optimal bound, our method allows for a broader range of learning rates with $\eta_{max}/\eta_{min} = O(\sqrt{T})$. This provides significantly greater flexibility in fine-tuning the learning rate parameter.

5. Experiments

In this section, we present the numerical performance of the CTMA and μ^2-SGD algorithms. We assess their robustness against *sign-flipping* and *label-flipping* attacks (Allen-Zhu et al., 2020), as well as state-of-the-art (SOTA) attacks, including the *empire* attack (Xie et al., 2020) and the *little* attack (Baruch et al., 2019). Additionally, we perform a comparative analysis of CTMA alongside two existing methods: Bucketing and NNM. Furthermore, we explore the effectiveness of employing a double meta-aggregator approach, where NNM serves as the initial meta-aggregator before passing through a robust aggregation phase, followed by the application of CTMA.

Our experiments also measure the performance of the μ^2-SGD algorithm compared to the standard momentum (Karimireddy et al., 2021), extending the evaluation to examine their resilience across a wide range of learning rates.

We conducted our experiments in a homogeneous setting. Specifically, we utilized the MNIST (LeCun et al., 2010) dataset, which contains 28x28 pixel grayscale images of handwritten digits, and the CIFAR-10 (Krizhevsky et al., 2014) dataset, which includes 32x32 color images spanning 10 classes. The detailed experimental setup and the complete results are provided in Appendix E. Our results demonstrate consistent outcomes across both the CIFAR-10 and MNIST datasets.

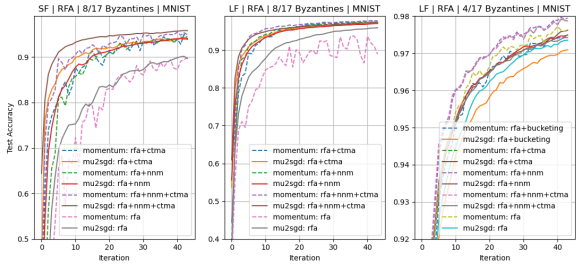
For the code, please visit our GitHub repository[1].

CTMA Versus Existing Meta Aggregators. Our results of CTMA show that it often performed as well as, and in some cases even better than, existing meta-aggregators such as Bucketing and NNM for both the momentum and μ^2-SGD algorithms. As illustrated in Figures 1(a) and 1(b), and demonstrated in Appendix E, CTMA enhances the performance of baseline robust-aggregators, exhibiting exceptional robustness and accuracy under challenging conditions. However, Figure 5 indicates that CTMA may struggle with heavy low-variance attacks, such as *little* and *empire*, which fall under the stochastic "noise blanket" and complicate the identification of adversarial workers, leading to performance degradation. Despite this, its computational efficiency and effectiveness against other realistic attacks, such as *sign-flipping* and *label-flipping*, as well as under weaker low-variance attacks (see Figures 6 and 9), make CTMA a valuable tool in the field of meta-aggregation.

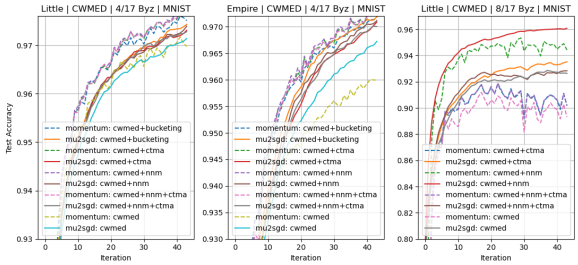
Additionally, integrating NNM with CTMA can even further improve performance. For instance, Figure 1(a) shows that in the case of the *sign-flipping* attack with Robust Aggregation for Federated Learning (RFA) (Pillutla et al., 2022) (an approximation of the GM aggregator), the combination of NNM and CTMA significantly improved both robustness and accuracy for both momentum and μ^2-SGD.

μ^2-SGD Versus Momentum. We compared the performance of the μ^2-SGD algorithm against the standard momentum across a wide range of learning rates and various attacks. The results in Figures 2(a) and 1(a) show that μ^2-SGD outperformed momentum under severe attacks when

[1] https://github.com/dahan198/synchronous-fault-tolerant-ml

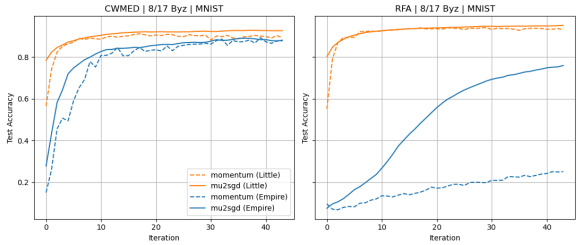


(a) Performance comparison of CTMA with existing meta-aggregators under *sign-flipping* and *label-flipping* attacks.

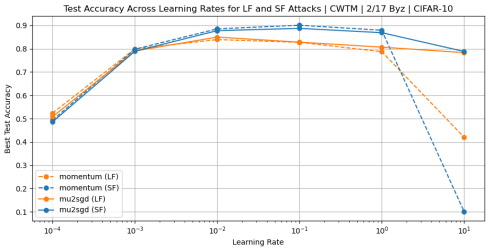


(b) Performance comparison of CTMA with existing meta-aggregators under *empire* and *little* attacks.

Figure 1. Performance comparison of CTMA with existing meta-aggregators (Conf. 1 in Table 2).



(a) Performance comparison of μ^2-SGD with standard momentum under *empire* and *little* attacks (Conf. 1 in Table 2).



(b) Performance comparison of μ^2-SGD with standard momentum for a wide range of learning rates, under *sign-flipping* and *label-flipping* attacks (Conf. 4 in Table 2).

Figure 2. Performance comparison of μ^2-SGD with standard momentum.

nearly half of the workers were Byzantine. Figure 2(a) highlights μ^2-SGD's strong resilience against heavy SOTA attacks like *little* and *empire*. In these cases, μ^2-SGD exhibited superior convergence speed and stability during training iterations by considering the entire history of gradients and query points, which aligns well with our theory. However, in cases of weaker attacks, as shown in Figure 1(a) for the *label-flipping* attack with 4 out of 17 Byzantine workers, or when a meta-aggregator is used to enhance performance, as depicted in Figure 1(b), the high stability of μ^2-SGD is not necessary. In fact, in these scenarios, the noisier momentum achieved faster convergence. This is because μ^2-SGD, with its double momentum mechanism and parameters that account for the entire history, takes very small and cautious steps, potentially slowing its convergence compared to a single momentum mechanism with a larger parameter.

Furthermore, Figure 2(b) shows that μ^2-SGD exhibited greater stability across a wide range of learning rates for *label-flipping* and *sign-flipping* attacks, whereas momentum tended to be more sensitive to this parameter. This provides a broader range of options for hyper-parameter optimization of μ^2-SGD.

Conclusions and Future Work

We introduce two complementary techniques to enhance the efficiency and practicality of Byzantine-robust training. First, we present a novel meta-aggregator that substantially improves efficiency compared to previous methods. Second, we incorporate a recent novel gradient estimation technique into Byzantine-robust training. We substantiate the theoretical benefits of our new approaches and corroborate them in practice, demonstrating an advantage over momentum, the standard workhorse of Byzantine-robust training.

Future directions: **(i)** extending our investigation to the challenging setting of decentralized Byzantine-robust training (He et al., 2022); **(ii)** understand whether we can further develop estimates that incorporate (stochastic) second order information in the spirit of Antonakopoulos et al. (2022); in order to yield an even more aggressive variance reduction, which will give rise to easier mitigation of faulty updates.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning, and especially the aspect of fault-tolerant and reliable training. There are many potential

societal consequences of our work, none of which we feel must be specifically highlighted here.

Acknowledgement

This research was partially supported by Israel PBC-VATAT, by the Technion Artificial Intelligent Hub (Tech.AI) and by the Israel Science Foundation (grant No. 447/20).

References

Acharya, A., Hashemi, A., Jain, P., Sanghavi, S., Dhillon, I. S., and Topcu, U. Robust training in high dimensions via block coordinate geometric median descent. In *International Conference on Artificial Intelligence and Statistics*, pp. 11145–11168. PMLR, 2022.

Alistarh, D., Allen-Zhu, Z., and Li, J. Byzantine stochastic gradient descent. *Advances in Neural Information Processing Systems*, 31, 2018.

Allen-Zhu, Z., Ebrahimian, F., Li, J., and Alistarh, D. Byzantine-resilient non-convex stochastic gradient descent. *arXiv preprint arXiv:2012.14368*, 2020.

Allouah, Y., Farhadkhani, S., Guerraoui, R., Gupta, N., Pinot, R., and Stephan, J. Fixing by mixing: A recipe for optimal byzantine ml under heterogeneity. In *International Conference on Artificial Intelligence and Statistics*, pp. 1232–1300. PMLR, 2023.

Antonakopoulos, K., Kavis, A., and Cevher, V. Extra-newton: A first approach to noise-adaptive accelerated second-order methods. *Advances in Neural Information Processing Systems*, 35:29859–29872, 2022.

Baruch, G., Baruch, M., and Goldberg, Y. A little is enough: Circumventing defenses for distributed learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Blanchard, P., El Mhamdi, E. M., Guerraoui, R., and Stainer, J. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in neural information processing systems*, 30, 2017.

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečnỳ, J., Mazzocchi, S., McMahan, B., et al. Towards federated learning at scale: System design. *Proceedings of machine learning and systems*, 1:374–388, 2019.

Chen, Y., Su, L., and Xu, J. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(2):1–25, 2017.

Cutkosky, A. Anytime online-to-batch, optimism and acceleration. In *International conference on machine learning*, pp. 1446–1454. PMLR, 2019.

Cutkosky, A. and Orabona, F. Momentum-based variance reduction in non-convex sgd. *Advances in neural information processing systems*, 32, 2019.

Dekel, O., Gilad-Bachrach, R., Shamir, O., and Xiao, L. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13(1), 2012.

El Mhamdi, E. M., Guerraoui, R., and Rouault, S. L. A. Distributed momentum for byzantine-resilient stochastic gradient descent. In *9th International Conference on Learning Representations (ICLR)*, number CONF, 2021.

Farhadkhani, S., Guerraoui, R., Gupta, N., Pinot, R., and Stephan, J. Byzantine machine learning made easy by resilient averaging of momentums. In *International Conference on Machine Learning*, pp. 6246–6283. PMLR, 2022.

Guerraoui, R., Rouault, S., et al. The hidden vulnerability of distributed learning in byzantium. In *International Conference on Machine Learning*, pp. 3521–3530. PMLR, 2018.

Guerraoui, R., Gupta, N., and Pinot, R. Byzantine machine learning: A primer. *ACM Computing Surveys*, 2023.

Hazan, E. et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

He, L., Karimireddy, S. P., and Jaggi, M. Byzantine-robust decentralized learning via clippedgossip. *arXiv preprint arXiv:2202.01545*, 2022.

Karimireddy, S. P., He, L., and Jaggi, M. Byzantine-robust learning on heterogeneous datasets via bucketing. *arXiv preprint arXiv:2006.09365*, 2020.

Karimireddy, S. P., He, L., and Jaggi, M. Learning from history for byzantine robust optimization. In *International Conference on Machine Learning*, pp. 5311–5319. PMLR, 2021.

Kavis, A., Levy, K. Y., Bach, F., and Cevher, V. Unixgrad: A universal, adaptive algorithm with optimal guarantees for constrained optimization. *Advances in neural information processing systems*, 32, 2019.

Krizhevsky, A., Nair, V., and Hinton, G. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55 (5), 2014.

Lamport, L., Shostak, R., and Pease, M. The byzantine generals problem. In *Concurrency: the works of leslie lamport*, pp. 203–226. 2019.

LeCun, Y., Cortes, C., Burges, C., et al. Mnist handwritten digit database, 2010.

Levy, K. Y. μ^2-sgd: Stable stochastic optimization via a double momentum mechanism. *arXiv preprint arXiv:2304.04172*, 2023.

Pillutla, K., Kakade, S. M., and Harchaoui, Z. Robust aggregation for federated learning. *IEEE Transactions on Signal Processing*, 70:1142–1154, 2022.

Shalev-Shwartz, S., Shamir, O., Srebro, N., and Sridharan, K. Stochastic convex optimization. In *COLT*, volume 2, pp. 5, 2009.

Xie, C., Koyejo, O., and Gupta, I. Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation. In *Uncertainty in Artificial Intelligence*, pp. 261–270. PMLR, 2020.

Yin, D., Chen, Y., Kannan, R., and Bartlett, P. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, pp. 5650–5659. PMLR, 2018.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

A. Bounded Smoothness Variance Assumption

We show that Eq. (4) implies that Eq. (5) holds for some $\sigma_L^2 \in [0, L^2]$.

$$\begin{aligned}\mathbb{E}\|(\nabla f_i(\mathbf{x};\mathbf{z}) - \nabla f_i(\mathbf{x})) - (\nabla f_i(\mathbf{y};\mathbf{z}) - \nabla f_i(\mathbf{y}))\|^2 &= \mathbb{E}\|\nabla f_i(\mathbf{x};\mathbf{z}) - \nabla f_i(\mathbf{y};\mathbf{z})\|^2 - \|\nabla f_i(\mathbf{x}) - \nabla f_i(\mathbf{y}))\|^2 \\ &\leq L^2\|\mathbf{x}-\mathbf{y}\|^2 \ .\end{aligned}$$

Here, we also used $\mathbb{E}(\nabla f_i(\mathbf{x};\mathbf{z}) - \nabla f_i(\mathbf{y};\mathbf{z})) = (\nabla f_i(\mathbf{x}) - \nabla f_i(\mathbf{y}))$, and followed Eq. (4). Therefore, we establish that $\sigma_L^2 \in [0, L^2]$.

B. Synchronous μ^2-SGD Analysis

B.1. Proof of Thm. 4.1

Proof of Thm. 4.1. We follow similar steps as in he proof of Theorem 4.1 of Levy (2023). First, we consider the definition of $\mathbf{x}_t$, which leads us to the following relationship:

$$\alpha_{1:t-1}(\mathbf{x}_t - \mathbf{x}_{t-1}) = \alpha_t(\mathbf{w}_t - \mathbf{x}_t) \ ,$$

which further yields to,

$$\|\mathbf{x}_t - \mathbf{x}_{t-1}\| = (\alpha_t/\alpha_{1:t-1})\|\mathbf{w}_t - \mathbf{x}_t\| \ .$$

Given that $\mathbf{w}_t$ belongs to $\mathcal{K}$ by its definition, and $\mathbf{x}_t$ is a weighted average of $\{\mathbf{w}_t\}_t$, the convexity of $\mathcal{K}$ ensures that $\mathbf{x}_t \in \mathcal{K}$. Furthermore, $\|\mathbf{w}_t - \mathbf{x}_t\| \leq D$ in accordance with the assumption in Eq. (1). Assigning $\alpha_t = t$, leads us to establish the ratio $\frac{\alpha_t}{\alpha_{1:t-1}} = \frac{2}{t-1}$. Consequently, for any $t \geq 1$,

$$\|\mathbf{x}_t - \mathbf{x}_{t-1}\| \leq \frac{2}{t-1}D \ . \tag{6}$$

We proceed to analyze the recursive dynamics of $\varepsilon_t^{(i)}$ for each $i \in \mathcal{G}$. Based on the definitions of $\mathbf{d}_t^{(i)}$ and $\varepsilon_t^{(i)}$, we can present the recursive relationship in the following way:

$$\varepsilon_t^{(i)} = \beta_t(\mathbf{g}_t^{(i)} - \nabla f_i(\mathbf{x}_t)) + (1-\beta_t)Z_t^{(i)} + (1-\beta_t)\varepsilon_{t-1}^{(i)} \ ,$$

where $Z_t^{(i)} := (\mathbf{g}_t^{(i)} - \nabla f_i(\mathbf{x}_t)) - (\tilde{\mathbf{g}}_{t-1}^{(i)} - \nabla f_i(\mathbf{x}_{t-1}))$. Upon choosing $\beta_t = \frac{1}{t}$, we can reformulate the equation as follows:

$$t\varepsilon_t^{(i)} = (\mathbf{g}_t^{(i)} - \nabla f_i(\mathbf{x}_t)) + (t-1)Z_t^{(i)} + (t-1)\varepsilon_{t-1}^{(i)} = \mathcal{M}_t^{(i)} + (t-1)\varepsilon_{t-1}^{(i)} \ ,$$

where $\mathcal{M}_t^{(i)} := (\mathbf{g}_t^{(i)} - \nabla f_i(\mathbf{x}_t)) + (t-1)Z_t^{(i)}$. Unrolling this recursion yields an explicit expression for any $t \in [T]$:

$$t\varepsilon_t^{(i)} = \sum_{\tau\in[t]} \mathcal{M}_\tau^{(i)} \ .$$

Following this, we derive an upper bound for the expected square norm of $\mathcal{M}_\tau^{(i)}$ as follows:

$$\begin{aligned}\mathbb{E}\|\mathcal{M}_\tau^{(i)}\|^2 &= \mathbb{E}\|(\mathbf{g}_\tau^{(i)} - \nabla f_i(\mathbf{x}_\tau)) + (\tau-1)Z_\tau^{(i)}\|^2 \\ &\leq 2\mathbb{E}\|\mathbf{g}_\tau^{(i)} - \nabla f_i(\mathbf{x}_\tau)\|^2 + 2\mathbb{E}\|(\tau-1)Z_\tau^{(i)}\|^2 \\ &= 2\mathbb{E}\|\mathbf{g}_\tau^{(i)} - \nabla f_i(\mathbf{x}_\tau)\|^2 + 2(\tau-1)^2\mathbb{E}\|(\mathbf{g}_\tau^{(i)} - \nabla f_i(\mathbf{x}_\tau)) - (\tilde{\mathbf{g}}_{\tau-1}^{(i)} - \nabla f_i(\mathbf{x}_{\tau-1}))\|^2 \\ &\leq 2\sigma^2 + 2\sigma_L^2(\tau-1)^2\mathbb{E}\|\mathbf{x}_\tau - \mathbf{x}_{\tau-1}\|^2 \\ &\leq 2\sigma^2 + 8D^2\sigma_L^2 = \tilde{\sigma}^2 \ ,\end{aligned}$$

where the first inequality uses $\|\mathbf{a}+\mathbf{b}\|^2 \leq 2\|\mathbf{a}\|^2 + 2\|\mathbf{b}\|^2$, which holds $\forall \mathbf{a}, \mathbf{b} \in \mathbb{R}^d$. The second inequality employs the assumptions outlined in Equations (2) and (5). The third inequality follows Eq. (6).

Note that for each $i \in \mathcal{G}$, the sequence $\{\mathcal{M}_t^{(i)}\}_t$ constitutes a martingale difference sequence relative to a natural filtration $\{\mathcal{F}_t^{(i)}\}_t$. Furthermore, the sequence $\{\mathcal{M}_t^{(i)}\}_{t,i}$ forms $|\mathcal{G}|$ independent martingale difference $\{\mathcal{M}_t^{(i)}\}_t$ sequences relative to a natural filtration $\mathcal{F}_t$.

Lemma B.1. *Consider $\{M_t^{(i)}\}_{t,i}$, a collection of m martingale difference sequences $\{M_t^{(i)}\}_t$ with a respect to a natural filtration $\mathcal{F}_t$. These sequences are independent for each $i \in [m]$ with a respect to natural filtration $\{\mathcal{F}_t^{(i)}\}_t$. Then, for any $t \geq 1$, the following holds:*

$$\mathbb{E}\left\|\sum_{i\in[m]}\sum_{\tau\in[t]} M_\tau^{(i)}\right\|^2 = \sum_{i\in[m]}\sum_{\tau\in[t]} \mathbb{E}\left\|M_\tau^{(i)}\right\|^2 .$$

Leveraging Lemma B.1, we get the following relationship:

$$\mathbb{E}\left\|t\sum_{i\in\mathcal{G}}\varepsilon_t^{(i)}\right\|^2 = \mathbb{E}\left\|\sum_{i\in\mathcal{G}}\sum_{\tau\in[t]}\mathcal{M}_\tau^{(i)}\right\|^2 = \sum_{i\in\mathcal{G}}\sum_{\tau\in[t]}\mathbb{E}\left\|\mathcal{M}_\tau^{(i)}\right\|^2 \leq \tilde{\sigma}^2 t|\mathcal{G}| .$$

In a similar way, for the individual error, we have:

$$\mathbb{E}\left\|t\varepsilon_t^{(i)}\right\|^2 = \mathbb{E}\left\|\sum_{\tau\in[t]}\mathcal{M}_\tau^{(i)}\right\|^2 = \sum_{\tau\in[t]}\mathbb{E}\left\|\mathcal{M}_\tau^{(i)}\right\|^2 \leq \tilde{\sigma}^2 t .$$

□

B.1.1. Proof of Lemma B.1

Proof of Lemma B.1.

Lemma B.2 (Borrowed from Lemma B.1 in Levy (2023))**.** *Let $\{M_t\}_t$ be a martingale difference sequence with respect to a filtration $\{\mathcal{F}_t\}_t$, then the following holds for any t,*

$$\mathbb{E}\left\|\sum_{\tau\in[t]} M_\tau\right\|^2 = \sum_{\tau\in[t]}\mathbb{E}\left\|M_\tau\right\|^2 .$$

For any $\tau \in [t]$, we have that,

$$\begin{aligned}
\mathbb{E}\left\|\sum_{i\in[m]} M_\tau^{(i)}\right\|^2 &= \sum_{i\in[m]}\mathbb{E}\left\|M_\tau^{(i)}\right\|^2 + \sum_{i\neq j;\ i,j\in[m]}\mathbb{E}\langle M_\tau^{(i)}, M_\tau^{(j)}\rangle \\
&= \sum_{i\in[m]}\mathbb{E}\left\|M_\tau^{(i)}\right\|^2 + \sum_{i\neq j;\ i,j\in[m]}\mathbb{E}\left[\underbrace{\mathbb{E}\left[\langle M_\tau^{(i)}, M_\tau^{(j)}\rangle|\mathcal{F}_{\tau-1}\right]}_{=0}\right] \\
&= \sum_{i\in[m]}\mathbb{E}\left\|M_\tau^{(i)}\right\|^2 , \qquad (7)
\end{aligned}$$

where the second equality is obtained by applying the law of total expectation. For the third equality, we rely on the independence of $M_\tau^{(i)}$ and $M_\tau^{(j)}$ for each $i, j \in [m]$, $i \neq j$ and the fact that $\{M_\tau^{(i)}\}_\tau$ constitutes a martingale difference sequence, which implies $\mathbb{E}\left[M_\tau^{(i)}\middle|\mathcal{F}_{\tau-1}\right] = 0$.

Therefore,

$$\mathbb{E}\left\|\sum_{i\in[m]}\sum_{\tau\in[t]} M_\tau^{(i)}\right\|^2 = \mathbb{E}\left\|\sum_{\tau\in[t]}\sum_{i\in[m]} M_\tau^{(i)}\right\|^2 = \sum_{\tau\in[t]}\mathbb{E}\left\|\sum_{i\in[m]} M_\tau^{(i)}\right\|^2 = \sum_{\tau\in[t]}\sum_{i\in[m]}\mathbb{E}\left\|M_\tau^{(i)}\right\|^2 ,$$

where the second equality is a result of applying Lemma B.2, which is appropriate since the sequence $\left\{\sum_{i\in[m]} M_t^{(i)}\right\}_t$ forms a martingale difference sequence. The third equality follows the result in Eq. (7). □

B.2. Proof of Thm. 4.3

Proof of Thm. 4.3. We begin by revisiting the AnyTime guarantee as outlined in Cutkosky (2019) and adapt the standard regret analysis of the update rule, as detailed in Hazan et al. (2016), to our context.

Theorem B.3 (Rephrased from Theorem 1 in Cutkosky (2019))**.** *Let $f : \mathcal{K} \to \mathbb{R}$ be a convex function with a minimum $\mathbf{x}^* \in \arg\min_{\mathbf{w}\in\mathcal{K}} f(\mathbf{w})$. Also let $\{\alpha_t \geq 0\}_t$, and $\{\mathbf{w}_t \in \mathcal{K}\}_t$, $\{\mathbf{x}_t \in \mathcal{K}\}_t$, such that $\{\mathbf{x}_t\}_t$ is an $\{\alpha_t\}_t$ weighted averaged of $\{\mathbf{w}_t\}_t$, i.e. such that $\mathbf{x}_1 = \mathbf{w}_1$, and for any $t \geq 1$,*

$$\mathbf{x}_{t+1} = \frac{1}{\alpha_{1:t+1}} \sum_{\tau\in[t+1]} \alpha_\tau \mathbf{w}_\tau \ .$$

Then the following holds for any $t \geq 1$:

$$\alpha_{1:t}(f(\mathbf{x}_t) - f(\mathbf{x}^*)) \leq \sum_{\tau\in[t]} \alpha_\tau \nabla f(\mathbf{x}_\tau)(\mathbf{w}_\tau - \mathbf{x}^*) \ .$$

Lemma B.4. *Let $f : \mathcal{K} \to \mathbb{R}$ be a convex function with a minimum $\mathbf{x}^* \in \arg\min_{\mathbf{w}\in\mathcal{K}} f(\mathbf{w})$, and assume that the assumption in Eq. (1) holds. Also let $\{\alpha_t \geq 0\}_t$, and $\{\mathbf{w}_t \in \mathcal{K}\}_t$. Then, for any $t \geq 1$, an arbitrary vector $\hat{\mathbf{d}}_t \in \mathbb{R}^d$, and the update rule:*

$$\mathbf{w}_{t+1} = \Pi_{\mathcal{K}} \left(\mathbf{w}_t - \eta\alpha_t \hat{\mathbf{d}}_t \right) \ ,$$

we have,

$$\sum_{\tau=1}^{t} \alpha_\tau \langle \hat{\mathbf{d}}_\tau, \mathbf{w}_{\tau+1} - \mathbf{x}^* \rangle \leq \frac{D^2}{2\eta} - \frac{1}{2\eta} \sum_{\tau=1}^{t} \|\mathbf{w}_\tau - \mathbf{w}_{\tau+1}\|^2 \ .$$

Lemma B.5. *let $f : \mathcal{K} \to \mathbb{R}$ be an L-smooth and convex function, and let $\mathbf{x}^* \in \arg\min_{\mathbf{x}\in\mathcal{K}} f(\mathbf{x})$, then for any $\mathbf{x} \in \mathbb{R}^d$ we have,*

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{x}^*)\|^2 \leq 2L(f(\mathbf{x}) - f(\mathbf{x}^*)) \ .$$

Next, for every iteration $t \leq T$, we define:

$$\hat{\mathbf{d}}_t := \mathcal{A}\left(\mathbf{d}_t^{(1)}, \ldots, \mathbf{d}_t^{(m)}\right) \ ,$$
$$\varepsilon_t := \hat{\mathbf{d}}_t - \nabla f(\mathbf{x}_t) \ .$$

Thus, combininh Theorem B.3 with Lemma B.4, we have that,

$$\begin{aligned}
\alpha_{1:t}(f(\mathbf{x}_t) - f(\mathbf{x}^*)) &\leq \sum_{\tau\in[t]} \alpha_\tau \langle \nabla f(\mathbf{x}_\tau), \mathbf{w}_\tau - \mathbf{x}^* \rangle \\
&= \sum_{\tau\in[t]} \alpha_\tau \langle \hat{\mathbf{d}}_\tau, \mathbf{w}_{\tau+1} - \mathbf{x}^* \rangle + \sum_{\tau\in[t]} \alpha_\tau \langle \hat{\mathbf{d}}_\tau, \mathbf{w}_\tau - \mathbf{w}_{\tau+1} \rangle - \sum_{\tau\in[t]} \alpha_\tau \langle \varepsilon_\tau, \mathbf{w}_\tau - \mathbf{x}^* \rangle \\
&\leq \frac{D^2}{2\eta} - \frac{1}{2\eta} \sum_{\tau\in[t]} \|\mathbf{w}_\tau - \mathbf{w}_{\tau+1}\|^2 + \sum_{\tau\in[t]} \alpha_\tau \langle \hat{\mathbf{d}}_\tau, \mathbf{w}_\tau - \mathbf{w}_{\tau+1} \rangle - \sum_{\tau\in[t]} \alpha_\tau \langle \varepsilon_\tau, \mathbf{w}_\tau - \mathbf{x}^* \rangle \\
&= \frac{D^2}{2\eta} - \frac{1}{2\eta} \sum_{\tau\in[t]} \|\mathbf{w}_\tau - \mathbf{w}_{\tau+1}\|^2 + \sum_{\tau\in[t]} \alpha_\tau \langle \nabla f(\mathbf{x}_\tau), \mathbf{w}_\tau - \mathbf{w}_{\tau+1} \rangle - \sum_{\tau\in[t]} \alpha_\tau \langle \varepsilon_\tau, \mathbf{w}_{\tau+1} - \mathbf{x}^* \rangle \\
&\leq \frac{D^2}{2\eta} \underbrace{- \frac{1}{2\eta} \sum_{\tau\in[t]} \|\mathbf{w}_\tau - \mathbf{w}_{\tau+1}\|^2 + \sum_{\tau\in[t]} \alpha_\tau \langle \nabla f(\mathbf{x}_\tau), \mathbf{w}_\tau - \mathbf{w}_{\tau+1} \rangle}_{\text{(A)}} + D \sum_{\tau\in[t]} \alpha_\tau \|\varepsilon_\tau\| \ , \qquad (8)
\end{aligned}$$

where the first inequality is derived from the Anytime guarantee, as outlined in Theorem B.3. The second inequality follows Lemma B.4. The third inequality is a result of applying the Cauchy-Schwarz inequality and the assumption in Eq. (1).

$$\begin{aligned}
\text{(A)} &:= -\frac{1}{2\eta}\sum_{\tau\in[t]}\|\mathbf{w}_\tau-\mathbf{w}_{\tau+1}\|^2+\sum_{\tau\in[t]}\alpha_\tau\langle\nabla f(\mathbf{x}_\tau),\mathbf{w}_\tau-\mathbf{w}_{\tau+1}\rangle\\
&= -\frac{1}{2\eta}\sum_{\tau\in[t]}\|\mathbf{w}_\tau-\mathbf{w}_{\tau+1}\|^2+\sum_{\tau\in[t]}\alpha_\tau\langle\nabla f(\mathbf{x}_\tau)-\nabla f(\mathbf{x}^*),\mathbf{w}_\tau-\mathbf{w}_{\tau+1}\rangle+\sum_{\tau\in[t]}\alpha_\tau\langle\nabla f(\mathbf{x}^*),\mathbf{w}_\tau-\mathbf{w}_{\tau+1}\rangle\\
&\le \frac{\eta}{2}\sum_{\tau\in[t]}\alpha_\tau^2\|\nabla f(\mathbf{x}_\tau)-\nabla f(\mathbf{x}^*)\|^2+\sum_{\tau\in[t]}\alpha_\tau\langle\nabla f(\mathbf{x}^*),\mathbf{w}_\tau-\mathbf{w}_{\tau+1}\rangle\\
&\le 2\eta L\sum_{\tau\in[t]}\alpha_{1:\tau}\Delta_\tau+\sum_{\tau\in[t]}(\alpha_\tau-\alpha_{\tau-1})\langle\nabla f(\mathbf{x}^*),\mathbf{w}_\tau\rangle-\alpha_t\langle\nabla f(\mathbf{x}^*),\mathbf{w}_{t+1}\rangle\\
&= 2\eta L\sum_{\tau\in[t]}\alpha_{1:\tau}\Delta_\tau+\sum_{\tau\in[t]}(\alpha_\tau-\alpha_{\tau-1})\langle\nabla f(\mathbf{x}^*),\mathbf{w}_\tau-\mathbf{w}_{t+1}\rangle\\
&\le 2\eta L\sum_{\tau\in[t]}\alpha_{1:\tau}\Delta_\tau+\sum_{\tau\in[t]}(\alpha_\tau-\alpha_{\tau-1})\|\nabla f(\mathbf{x}^*)\|\|\mathbf{w}_\tau-\mathbf{w}_{t+1}\|\\
&\le \frac{1}{2T}\sum_{\tau\in[T]}\alpha_{1:T}\Delta_\tau+\alpha_t G^* D\,.
\end{aligned}$$

Here, the first inequality employs the Young's inequality. For the second inequality, we introduce the notation $\Delta_t := f(\mathbf{x}_t)-f(\mathbf{x}^*)$, and we follow Lemma B.5, which relates to the smoothness of the function f. In this step, we also set $\alpha_0=0$ and utilizes the property $\alpha_\tau^2\le 2\alpha_{1:t}$, given that $\alpha_\tau=\tau$. The third inequality uses the Cauchy-Schwarz inequality. The last inequality follows the assumption in Eq. (1). It uses the fact that $t\le T$ and $\Delta_t\ge 0, \forall t$. This step also incorporates the choice of an appropriate learning rate parameter $\eta\le 1/4LT$.

Plugging (A) into Eq. (8), gives us,

$$\alpha_{1:t}\Delta\le\frac{1}{2T}\sum_{\tau\in[T]}\alpha_{1:T}\Delta_\tau+\frac{D^2}{2\eta}+\alpha_t G^* D+D\sum_{\tau\in[t]}\alpha_\tau\|\varepsilon_\tau\|\,. \tag{9}$$

Lemma B.6 (Lemma C.2 in Levy (2023))**.** *let $\{A_t\}_{t\in[T]},\{B_t\}_{t\in[T]}$ be sequences of non-negative elements, and assume that for any $t\le T$,*

$$A_t\le B_T+\frac{1}{2T}\sum_{t\in[T]}A_t\,.$$

Then the following bound holds,

$$A_T\le 2B_T\,.$$

In the next step, let us define two terms: $A_t := \alpha_{1:t}\mathbb{E}\left[f(\mathbf{x}_t)-f(\mathbf{x}^*)\right]$ and $B_t := \frac{D^2}{2\eta}+\alpha_t G^* D+D\sum_{\tau\in[t]}\alpha_\tau\mathbb{E}\|\varepsilon_\tau\|$. Note that the series $\{B_t\}_t$ forms a non-decreasing series of non-negative values, implying $B_t\le B_T$ for any $t\in[T]$. As a result of Eq. (9), we have that $A_t\le B_T+\frac{1}{2T}\sum_{\tau\in[T]}A_\tau$.

By employing Lemma 4.2, we have,

$$\begin{aligned}
\sum_{\tau\in[T]}\alpha_\tau^2\mathbb{E}\|\varepsilon_\tau\|^2 &\le \sum_{\tau\in[T]}\alpha_\tau^2\left(\frac{4\tilde{\sigma}^2}{\tau m}+\frac{12c_\delta\tilde{\sigma}^2}{\tau}+6c_\delta\xi^2\right)\\
&= \left(\frac{4\tilde{\sigma}^2}{m}+12c_\delta\tilde{\sigma}^2\right)\sum_{\tau\in[T]}\tau+6c_\delta\xi^2\sum_{\tau\in[T]}\tau^2\\
&\le T^2\left(\frac{4\tilde{\sigma}^2}{m}+12c_\delta\tilde{\sigma}^2\right)+T^3 6c_\delta\xi^2.
\end{aligned}\tag{10}$$

Utilizing Jensen's inequality, enables us to establish,

$$\begin{aligned}\sum_{\tau\in[T]}\alpha_\tau\mathbb{E}\|\varepsilon_\tau\| &= \sum_{\tau\in[T]}\sqrt{(\alpha_\tau\mathbb{E}\|\varepsilon_\tau\|)^2} \leq \sqrt{T}\sqrt{\sum_{\tau\in[T]}\alpha_\tau^2\mathbb{E}\|\varepsilon_\tau\|^2} \\ &\leq \sqrt{T}\sqrt{T^2\left(\frac{4\tilde{\sigma}^2}{m}+12c_\delta\tilde{\sigma}^2\right)+T^3 6c_\delta\xi^2} \\ &\leq T^{1.5}\tilde{\sigma}\sqrt{\frac{4}{m}+12c_\delta}+6T^2\xi\sqrt{c_\delta}. \end{aligned} \tag{11}$$

where the first inequality employs Eq. (10) and the second inequality uses $\sqrt{a+b}\leq\sqrt{a}+\sqrt{b}$ for non-negative $a,b\in\mathbb{R}$. The explanation behind this can be seen through the following steps: $\left(\sqrt{a}+\sqrt{b}\right)^2 = a+2\sqrt{ab}+b\geq a+b$, whereby taking the square root of both sides of this equation, we obtain the desired inequality.

Leveraging Lemma B.6 and Eq. (11), and acknowledging that $\alpha_{1:T}=\Theta(T^2)$, as $\alpha_t=t$, it follows that:

$$\begin{aligned}\mathbb{E}[f(\mathbf{x}_T)-f(\mathbf{x}^*)] &\leq \frac{1}{T^2}\mathcal{B}_T \\ &= \frac{D^2}{2T^2\eta}+\frac{G^*D}{T}+\frac{D}{T^2}\sum_{\tau\in[T]}\alpha_\tau\mathbb{E}\|\varepsilon_\tau\| \\ &\leq \frac{D^2}{2T^2\eta}+\frac{G^*D}{T}+\frac{D\tilde{\sigma}}{\sqrt{T}}\sqrt{12c_\delta+\frac{4}{m}}+6D\xi\sqrt{c_\delta}\,. \end{aligned} \tag{12}$$

Finally, choosing the optimal $\eta\leq\frac{1}{4TL}$ gives us:

$$\mathbb{E}[f(\mathbf{x}_T)-f(\mathbf{x}^*)]\leq O\left(\frac{G^*D+LD^2}{T}+\frac{D\tilde{\sigma}}{\sqrt{T}}\sqrt{c_\delta+\frac{1}{m}}+\sqrt{c_\delta}D\xi\right)\,.$$

□

Learning Rate Range for Optimal Convergence To simplify our analysis, we assume G^* is negligible, which is a reasonable assumption. For example, if $\mathbf{x}^*=\arg\min_{\mathbf{x}\in\mathcal{K}}f(\mathbf{x})=\arg\min_{\mathbf{x}\in\mathbb{R}^d}f(\mathbf{x})$, then $\nabla f(\mathbf{x}^*)=0$, resulting in $G^*=0$. Considering that G^* is negligible and using Eq. (12), we derive the range for the learning rate η to maintain optimal convergence:

$$\frac{D^2}{2T^2\eta}\leq\frac{D\tilde{\sigma}}{\sqrt{T}}\sqrt{12c_\delta+\frac{4}{m}}+6D\xi\sqrt{c_\delta}\,.$$

Thus, the lower bound for η is:

$$\eta\geq\frac{D^2}{2T^2}\cdot\frac{1}{\frac{D\tilde{\sigma}}{\sqrt{T}}\sqrt{12c_\delta+\frac{4}{m}}+6D\xi\sqrt{c_\delta}}=\frac{D}{2\tilde{\sigma}T^{1.5}\sqrt{12c_\delta+\frac{4}{m}}+12\xi T^2}\,.$$

Given that $\eta\leq\frac{1}{4TL}$, the range for η is:

$$\eta\in\left[\frac{D}{2\tilde{\sigma}T^{1.5}\sqrt{12c_\delta+\frac{4}{m}}+12\xi T^2},\frac{1}{4TL}\right]\,. \tag{13}$$

This approach allows us to employ a broad spectrum of learning rates $\eta\in[\eta_{min},\eta_{max}]$, such that $\eta_{max}/\eta_{min}=O(\sqrt{T})$. In contrast, the standard momentum method (Karimireddy et al., 2020) requires a low-range of learning rates such that $\eta_{max}/\eta_{min}=O(1)$ to achieve an order optimal bound. Specifically, for a starting point $\mathbf{x}_0\in\mathbb{R}^d$ and $\mathbf{x}^*:=\arg\min_{\mathbf{x}\in\mathbb{R}^d}f(\mathbf{x})$, the learning rate for momentum with adaptation to our notations is given by:

$$\eta_{momentum}\simeq\min\left\{O\left(\sqrt{\frac{L(f(\mathbf{x}_0)-f(\mathbf{x}^*))+c_\delta(\xi^2+\sigma^2)}{TL^2\sigma^2(\frac{1}{m}+c_\delta)}}\right),\frac{1}{8L}\right\}\,.$$

B.2.1. Proof of Lemma B.4

Proof of Lemma B.4. The update rule $\mathbf{w}_{\tau+1} = \Pi_{\mathcal{K}}(\mathbf{w}_\tau - \eta\alpha_\tau\hat{\mathbf{d}}_\tau)$ can be expressed as a convex optimization problem within the set $\mathcal{K}$:

$$\mathbf{w}_{\tau+1} = \Pi_{\mathcal{K}}\left(\mathbf{w}_\tau - \eta\alpha_\tau\hat{\mathbf{d}}_\tau\right) = \arg\min_{\mathbf{w}\in\mathcal{K}} \|\mathbf{w}_\tau - \eta\alpha_\tau\hat{\mathbf{d}}_\tau - \mathbf{w}\|^2 = \arg\min_{\mathbf{w}\in\mathcal{K}}\{\alpha_\tau\langle\hat{\mathbf{d}}_\tau, \mathbf{w}-\mathbf{w}_\tau\rangle + \frac{1}{2\eta}\|\mathbf{w}-\mathbf{w}_\tau\|^2\} .$$

Here, the first equality is derived from the definition of the update rule, the second stems from the property of projection, and the final equality is obtained by reformulating the optimization problem in a way that does not affect the minimum value.

Given that $\mathbf{w}_{\tau+1}$ is the optimal solution of the above convex problem, by the optimality conditions, we have that:

$$\left\langle \alpha_\tau\hat{\mathbf{d}}_\tau + \frac{1}{\eta}(\mathbf{w}_{\tau+1} - \mathbf{w}_\tau), \mathbf{w} - \mathbf{w}_{\tau+1}\right\rangle \geq 0, \quad \forall \mathbf{w}\in\mathcal{K} .$$

Rearranging this, summing over $t \geq 1$ iterations, and taking $\mathbf{w} = \mathbf{x}^*$, we derive:

$$\begin{aligned}
\sum_{\tau\in[t]} \alpha_\tau\langle\hat{\mathbf{d}}_\tau, \mathbf{w}_{\tau+1} - \mathbf{x}^*\rangle &\leq \frac{1}{\eta}\sum_{\tau\in[t]}\langle\mathbf{w}_\tau - \mathbf{w}_{\tau+1}, \mathbf{w}_{\tau+1} - \mathbf{x}^*\rangle \\
&= \frac{1}{2\eta}\sum_{\tau\in[t]}\left(\|\mathbf{w}_\tau - \mathbf{x}^*\|^2 - \|\mathbf{w}_{\tau+1} - \mathbf{x}^*\|^2 - \|\mathbf{w}_\tau - \mathbf{w}_{\tau+1}\|^2\right) \\
&= \frac{1}{2\eta}\left(\|\mathbf{w}_1 - \mathbf{x}^*\|^2 - \|\mathbf{w}_{t+1} - \mathbf{x}^*\|^2 - \sum_{\tau\in[t]}\|\mathbf{w}_\tau - \mathbf{w}_{\tau+1}\|^2\right) \\
&\leq \frac{D^2}{2\eta} - \frac{1}{2\eta}\sum_{\tau=1}^{t}\|\mathbf{w}_\tau - \mathbf{w}_{\tau+1}\|^2 .
\end{aligned}$$

The first equality equality is achieved through algebraic manipulation, and the last inequality follows the assumption in Eq. (1). □

B.2.2. Proof of Lemma B.5

Proof of Lemma B.5.

Lemma B.7 (Lemma C.1 in Levy (2023))**.** *let $f : \mathbb{R}^d \to \mathbb{R}$ be an L-smooth function with a global minimum $\mathbf{w}^*$, then for any $\mathbf{w} \in \mathbb{R}^d$ we have,*

$$\|\nabla f(\mathbf{x})\|^2 \leq 2L(f(\mathbf{x}) - f(\mathbf{w}^*)) .$$

Let us define the function $h(\mathbf{x}) = f(\mathbf{x}) - f(\mathbf{x}^*) - \langle\nabla f(\mathbf{x}^*), \mathbf{x} - \mathbf{x}^*\rangle$. Given the convexity of $f(\mathbf{x})$, we have $f(\mathbf{x}) - f(\mathbf{x}^*) \geq \langle\nabla f(\mathbf{x}^*), \mathbf{x} - \mathbf{x}^*\rangle$, leading to $h(\mathbf{x}) \geq 0$. As $h(\mathbf{x}^*) = 0$, this implies that $\mathbf{x}^*$ is the global minimum of h. Applying Lemma B.7, gives us,

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{x}^*)\|^2 = \|\nabla h(\mathbf{x})\|^2 \leq 2L(f(\mathbf{x}) - f(\mathbf{x}^*)) .$$

□

C. Robust Aggregators Analysis

Definition C.1 (Rephrased from Allouah et al. (2023))**.** (κ, δ)**-robustness**. Let $\frac{|\mathcal{B}|}{m} = \delta < 1/2$ and $\kappa \geq 0$. An aggregation rule $\mathcal{A}$ is called (κ, δ)-robustness if for any m vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m$, and any set $S \subseteq [m]$ such that $|S| = |\mathcal{G}|$, we have,

$$\|\hat{\mathbf{x}} - \bar{\mathbf{x}}_S\|^2 \leq \frac{\kappa}{|S|}\sum_{i\in S}\|\mathbf{x}_i - \bar{\mathbf{x}}_S\|^2 ,$$

where $\bar{\mathbf{x}}_S := \frac{1}{|S|}\sum_{i\in S}\mathbf{x}_i$.

Lemma C.2. *Let $\hat{\mathbf{x}}$ be (κ, δ)-robustness where $\delta = \frac{|\mathcal{B}|}{m}$. We define (κ, δ)-robustness as a rephrased version of the robust concept originally introduced in Allouah et al. (2023) and outlined in Definition C.1. Then, $\hat{\mathbf{x}}$ is (κ, δ)-robust.*

Proof. Given that $\hat{\mathbf{x}}$ is (κ, δ)-robustness, by choosing $S = \mathcal{G}$, we can deduce that:

$$\mathbb{E}\|\hat{\mathbf{x}} - \bar{\mathbf{x}}_{\mathcal{G}}\|^2 \leq \frac{\kappa}{|\mathcal{G}|} \sum_{i \in \mathcal{G}} \mathbb{E}\|\mathbf{x}_i - \bar{\mathbf{x}}_{\mathcal{G}}\|^2 \leq \frac{\kappa}{|\mathcal{G}|} \sum_{i \in \mathcal{G}} \rho_i^2 = \kappa \rho^2 \ ,$$

where the second inequality follows the notations in Definition 3.1. □

D. μ^2-SGD Overview

Our approach adopts the μ^2-SGD algorithm (Levy, 2023), a novel method that combines two distinct momentum-based mechanisms for enhancing variance reduction.

AnyTime-SGD. The first mechanism originates from the AnyTime-SGD algorithm (Cutkosky, 2019). This algorithm employs a learning rate $\eta > 0$ and a series of non-negative weights $\{\alpha_t\}_t$. It operates by maintaining two series of query points, $\{\mathbf{w}_t\}_t$ and $\{\mathbf{x}_t\}_t$, and initializes $\mathbf{x}_1 = \mathbf{w}_1$. At each step t, the algorithm first updates $\mathbf{w}_{t+1}$ in a manner akin to the standard SGD, but adjusted by a factor of α_t for the gradient $\mathbf{g}_t := \nabla f(\mathbf{x}_t; \mathbf{z}_t)$. Subsequently, it calculates the next query point $\mathbf{x}_{t+1}$ through a weighted average of the accumulated query points up to step $t+1$, as shown in the following equation:

$$\mathbf{w}_{t+1} = \Pi_{\mathcal{K}}\left(\mathbf{w}_t - \eta \alpha_t \mathbf{g}_t\right), \quad \mathbf{x}_{t+1} = \frac{1}{\alpha_{1:t+1}} \sum_{k \in [t+1]} \alpha_k \mathbf{w}_k \ .$$

A fundamental characteristic of the AnyTime-SGD is that the query point $\mathbf{x}_T$ also serves as the algorithm's output. This contrasts with standard SGD, which typically outputs the average of all query points or randomly selects one. Thus, at any given iteration t, AnyTime-SGD consistently offers a potential solution for the optimization problem.

STORM. While the first mechanism employs weighted averaging of query points, the second leverages weighted averaging of the stochastic gradient estimators. This second mechanism is derived from the Stochastic Recursive Momentum (STORM) approach (Cutkosky & Orabona, 2019). It utilizes a corrected momentum technique, which serves as an estimator for the actual gradient. This correction is implemented by adding a bias to the standard momentum equation, thereby refining the gradient estimates from previous iterations. This method is defined as follows:

$$\mathbf{d}_t = \nabla f(\mathbf{x}_t; \mathbf{z}_t) + (1 - \beta_t)(\mathbf{d}_{t-1} - \nabla f(\mathbf{x}_{t-1}; \mathbf{z}_t)) \ .$$

A key feature of this approach is its ability to achieve implicit variance reduction through the use of the corrected momentum. This method not only adjusts the gradient estimations but also enhances the overall stability.

μ^2-SGD. The μ^2-SGD algorithm integrates the AnyTime update step with the STORM gradient estimator. Instead of using the stochastic gradient $\mathbf{g}_t$, it updates query points using a corrected momentum $\mathbf{d}_t$. The update process is defined as follows:

$$\mathbf{w}_{t+1} = \Pi_{\mathcal{K}}\left(\mathbf{w}_t - \eta \alpha_t \mathbf{d}_t\right), \ \mathbf{x}_{t+1} = \frac{1}{\alpha_{1:t+1}} \sum_{k \in [t+1]} \alpha_k \mathbf{w}_k, \ \mathbf{x}_1 = \mathbf{w}_1 \ . \tag{14}$$

A primary aspect of the μ^2-SGD algorithm is achieved by setting the momentum weights $\{\beta_t := 1/t\}_t$ and the AnyTime weights $\{\alpha_t := t\}_t$. This enables μ^2-SGD to significantly reduce the stochastic variance $\mathbb{E}\|\varepsilon_t\|^2 := \mathbb{E}\|\mathbf{d}_t - \nabla f(\mathbf{x}_t)\|^2 \leq O(\tilde{\sigma}^2/t)$ at step t, where $\tilde{\sigma}^2 \leq O(\sigma^2 + D^2\sigma_L^2)$. This means that the variance decreases with each iteration, eventually becoming negligible, contrasting with standard SGD where the variance $\mathbb{E}\|\varepsilon_t^{\text{SGD}}\|^2 := \mathbb{E}\|\mathbf{g}_t - \nabla f(\mathbf{x}_t)\|^2$ remains uniformly bounded. Moreover, by choosing $\beta_t = 1/t$, μ^2-SGD considers the entire gradient history (Karimireddy et al., 2021), aligning with the AnyTime update's consideration of the entire history of query points. This approach differs from other momentum methods (Karimireddy et al., 2021; 2020; Allouah et al., 2023), where the momentum weight β_t is tied to the learning rate η_t, thereby limiting the gradient history consideration to just $\sqrt{t}$ gradients.

E. Experiments

E.1. Technical Details

Remark E.1. Recall that the AnyTime update step of $\mathbf{x}_t$ is defined as:

$$\mathbf{x}_t := \frac{\alpha_t \mathbf{w}_t + \alpha_{1:t-1}\mathbf{x}_{t-1}}{\alpha_{1:t}}.$$

This formulation provides an alternative representation of the update step, which can be expressed as:

$$\mathbf{x}_t = \gamma_t \mathbf{w}_t + (1-\gamma_t)\mathbf{x}_{t-1},$$

where $\gamma_t := \frac{\alpha_t}{\alpha_{1:t}}$. Furthermore, by setting $\alpha_t = C\alpha_{1:t-1}$ for a constant $C > 0$, it follows that $\gamma_t = \frac{C\alpha_{1:t-1}}{C\alpha_{1:t-1}+\alpha_{1:t-1}} = \frac{C}{C+1}$ is a constant for every $t \geq 1$.

E.1.1. Datasets

We evaluated our approach on two benchmark datasets: CIFAR-10 (Krizhevsky et al., 2014) and MNIST (LeCun et al., 2010).

- **CIFAR-10**: This dataset consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class. The dataset is divided into 50,000 training images and 10,000 testing images.
- **MNIST**: This dataset consists of 70,000 28x28 grayscale images of handwritten digits in 10 classes, with 60,000 training images and 10,000 testing images.

E.1.2. Model Architectures

To demonstrate the efficiency of momentum parameters that account for the entire history, we conducted experiments using simple convolutional networks. Our choice of simple conv networks was driven by the need to mitigate the sensitivity to numerical errors, which is often encountered in more complex models with very small momentum parameters. Additionally, we extended our evaluation to a more complex scenario by using the CIFAR-10 dataset with a ResNet18 model, employing fixed momentum parameters for comparison.

- **Simple Conv Network for MNIST**: This model consists of two convolutional layers with batch normalization and ReLU activation, followed by max pooling layers. The first convolutional layer has 16 filters, and the second layer has 32 filters. The output from the convolutional layers is flattened and passed through a fully connected layer with 1,568 units for classification.
- **Simple Conv Network for CIFAR-10**: This model also consists of two convolutional layers with batch normalization and ReLU activation, followed by max pooling layers. The first convolutional layer has 16 filters, and the second layer has 32 filters. The output from the convolutional layers is flattened and passed through a fully connected layer with 2,048 units for classification.
- **ResNet18 for CIFAR-10**: We used the standard ResNet18 architecture, which includes multiple residual blocks with convolutional layers, batch normalization, and ReLU activation. This model is designed to handle more complex image classification tasks.

E.1.3. Algorithms and Baselines

We evaluate our proposed μ^2-SGD algorithm by comparing it with the standard momentum method as described by Karimireddy et al. (2021). The specific momentum parameters utilized in our experiments are detailed below:

- **Standard Momentum**: We employ $\beta_t = 0.9$, as suggested by Karimireddy et al. (2021).
- **μ^2-SGD**: We experiment with two distinct parameter settings:
  - Dynamic parameters: $\alpha_t = t$ and $\beta_t = 1/t$, in line with our theoretical suggestion.
  - Fixed parameters: $\gamma_t = 0.1$ and $\beta_t = 0.9$, more similar to the fixed standard momentum parameter.

  Detailed configurations are provided in Section E.1.5.

E.1.4. Evaluation Metrics

To assess the performance of our algorithm, we used the following metric:

- **Accuracy**: The proportion of correctly classified instances over the total instances.

E.1.5. Experimental Setup

We conducted a hyperparameter search to identify the optimal settings for our experiments.

Learning Rate: We experimented with a range of learning rates from 10^{-4} to 10^{1}. For experiments requiring a single learning rate, we selected 0.1, which was found to be optimal within this range.

The table below summarizes the configurations used in our experiments, including the settings for α_t, β_t, and γ_t for the μ^2-SGD algorithm, as well as the dataset, model, batch size, and gradient clipping values to enhance performance, as implemented in Allouah et al. (2023). Note that gradient clipping was not applied in the experiments involving a wide range of learning rates due to its impact on the size of the update step.

Configuration	α_t	β_t	γ_t	Dataset	Model	Batch Size	Gradient Clipping
1	t	$1/t$	-	MNIST	Simple Conv	4	2
2	t	$1/t$	-	CIFAR-10	Simple Conv	64	5
3	-	0.9	0.1	MNIST	Simple Conv	64	-
4	-	0.9	0.1	CIFAR-10	ResNet18	8	-

Table 2. Experimental Configurations

E.1.6. Attacks

To evaluate the robustness of our algorithms, we tested them against the following adversarial attacks:

- **Label-Flipping** (Allen-Zhu et al., 2020): This attack flips the original target labels to incorrect labels by subtracting the original label from 9,
$$\text{flipped_label} = 9 - \text{original_label}.$$
- **Sign-Flipping** (Allen-Zhu et al., 2020): This attack flips the signs of the momentums, in the spirit of faults that occur when bits are transmitted incorrectly from workers to the central server.
$$\text{Byzantine_update} = -\text{worker_momentum}.$$
- **A Little Is Enough** (Baruch et al., 2019): This attack is designed to lie under the stochastic "noise blanket." It calculates the maximum allowable deviation $z_{\max}$ based on the number of honest workers, then perturbs the honest updates by subtracting the product of the standard deviation and $z_{\max}$ from the mean of the honest updates.
$$\text{Byzantine_update} = \text{mean}(\text{honest_momentums}) - \text{std}(\text{honest_momentums}) \cdot z_{\max}.$$
- **Empire** (Xie et al., 2020): This attack scales the mean of the honest momentums by a small factor ϵ in the negative direction,
$$\text{Byzantine_update} = -\epsilon \cdot \text{mean}(\text{honest_momentums}).$$
where the mean and standard deviation are calculated coordinate-wise, and we set $\epsilon = 0.5$.

E.1.7. Implementation Details

The implementation was carried out using PyTorch. The code was written in Python and executed on NVIDIA A30 GPU for MNIST and NVIDIA GeForce RTX 3090 GPU for CIFAR-10. All experiments were repeated three times with different random seeds to ensure statistical significance, and the results reported are the averages of these runs.

E.2. Complete Experimental Results

E.2.1. MNIST

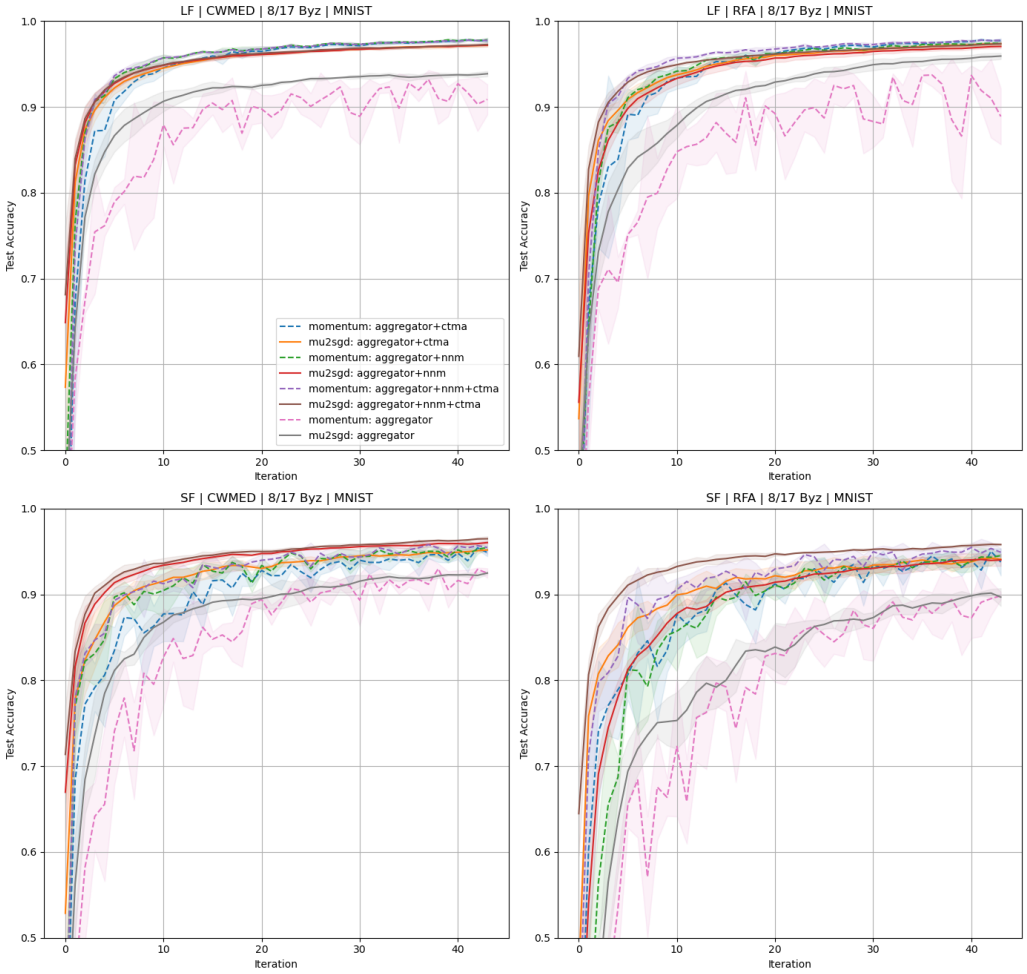


Figure 3. Performance comparison of CTMA with existing meta-aggregators, and μ^2-SGD with momentum under *sign-flipping* and *label-flipping* attacks with 8/17 Byzantine workers on the MNIST dataset (Conf. 1 in Table 2). Here, we observe that CTMA enhances the performance of both μ^2-SGD and momentum. Furthermore, CTMA performs at least as well as, if not better than, other meta-aggregators. The integration of nnm with CTMA can further improve performance. Notably, μ^2-SGD demonstrates high stability compared to momentum, even without the assistance of any meta-aggregator. This stability is valuable for increasing resilience against heavy attacks and boosting overall performance.

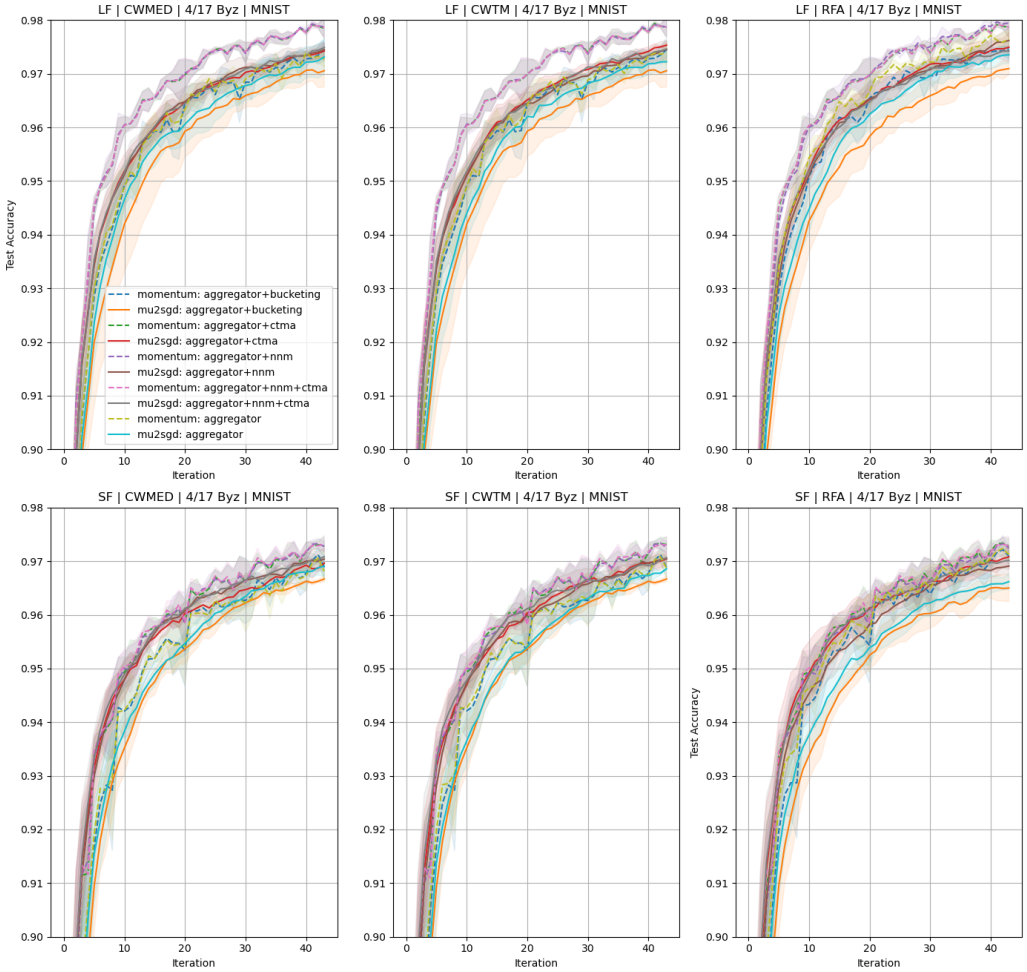


Figure 4. Performance comparison of CTMA with existing meta-aggregators, and μ^2-SGD with momentum under a weaker *sign-flipping* and *label-flipping* attack with 4/17 Byzantine workers on the MNIST dataset (Conf. 1 in Table 2). In this scenario, even though the performance of momentum is noisier over iterations compared to μ^2-SGD, it does not require additional stability to perform effectively and outperforms μ^2-SGD. CTMA enhances the performance of both μ^2-SGD and momentum, maintaining consistent improvement as observed under the heavier attacks shown in Figure 3.

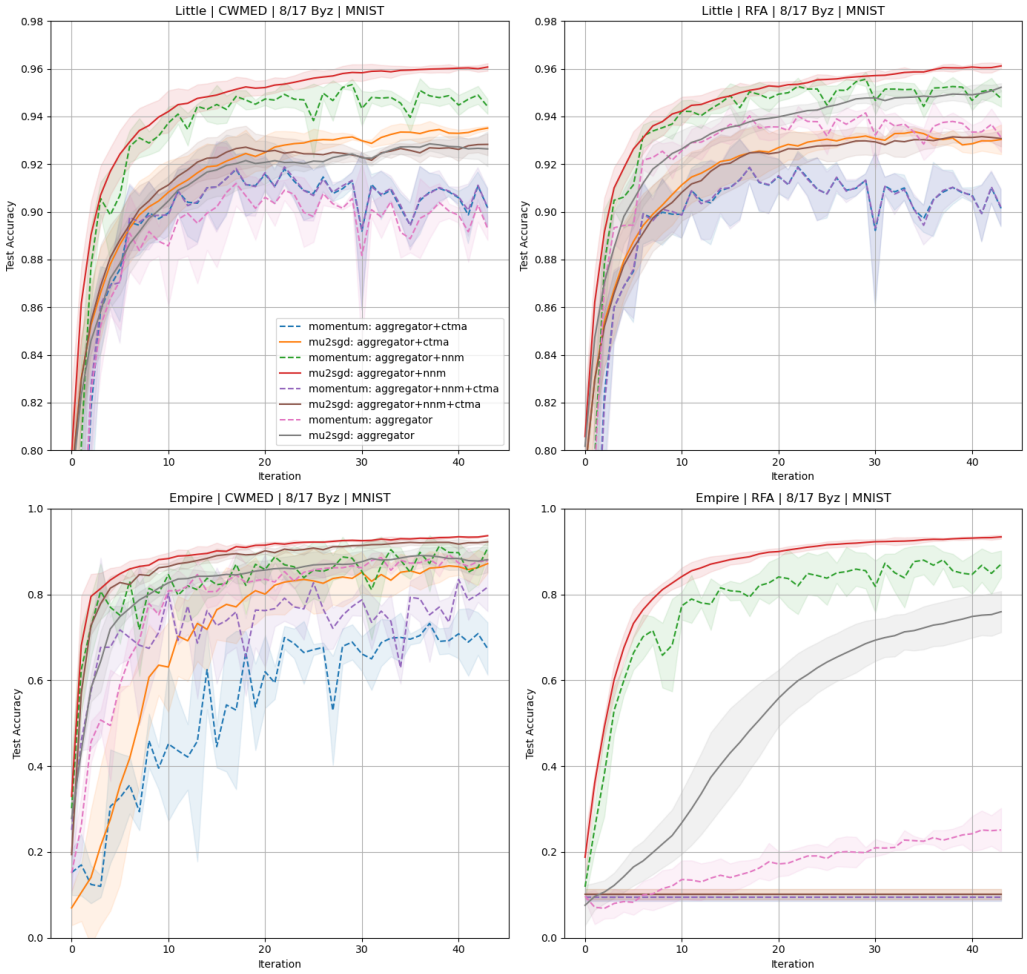


Figure 5. Performance comparison of CTMA with existing meta-aggregators, and μ^2-SGD with momentum under SOTA low-variance attacks, *empire* and *little*, with 8/17 Byzantine workers on the MNIST dataset (Conf. 1 in Table 2). These low-variance attacks are harder to detect and represent an especially severe attack scenario. In this context, CTMA performs poorly due to its strong reliance on the variance among the workers' outputs. In contrast, NNM performs very effectively for both momentum and μ^2-SGD. The low variance in μ^2-SGD enhances the effectiveness of NNM, making μ^2-SGD more robust and particularly valuable against heavy low-variance attacks compared to momentum, with or without the addition of NNM.

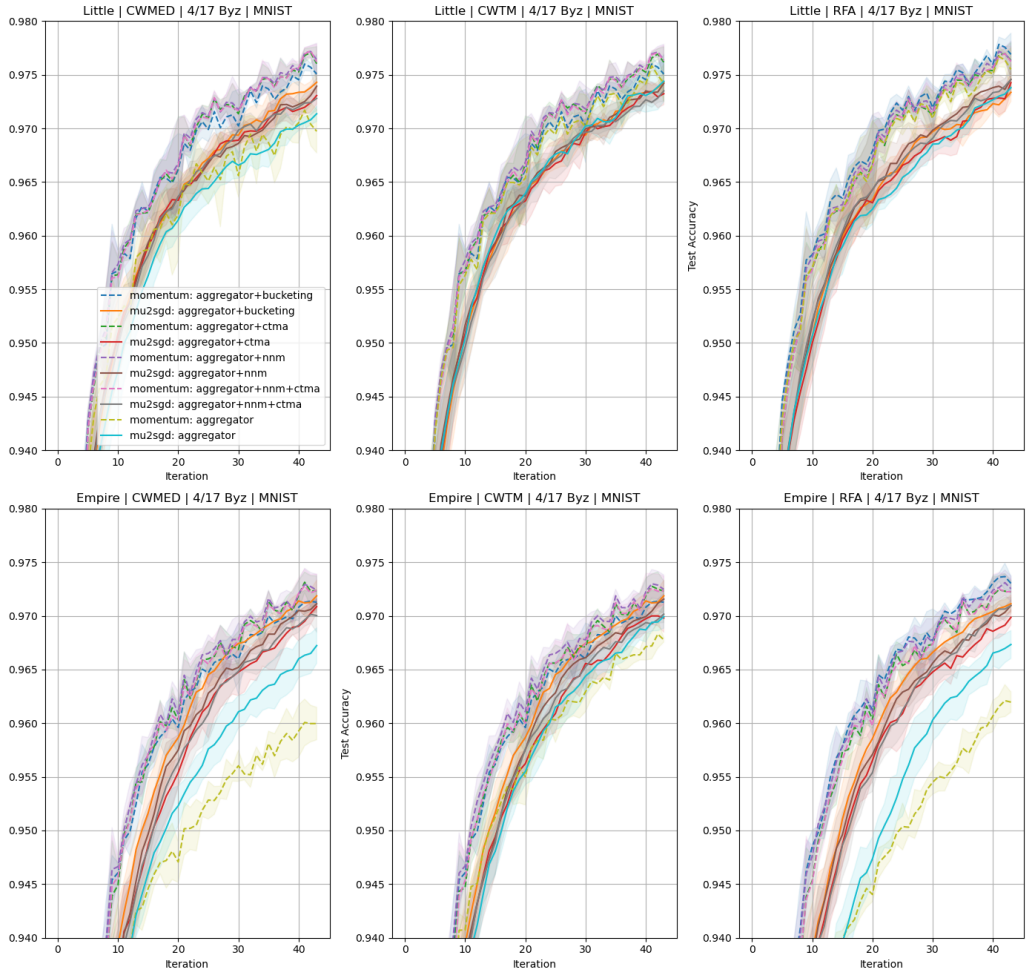


Figure 6. Performance comparison of CTMA with existing meta-aggregators, and μ^2-SGD with momentum under SOTA low-variance attacks, *empire* and *little*, with 4/17 Byzantine workers on the MNIST dataset (Conf. 1 in Table 2). These are low-variance attacks in a weaker scenario with 4 Byzantine workers. In this context, CTMA performs well and often compares favorably to other meta-aggregators, enhancing the performance of both μ^2-SGD and momentum. Furthermore, for the *empire* attack, the μ^2-SGD outperforms momentum without the addition of a meta-aggregator. However, when a meta-aggregator is added, the inherent stability of μ^2-SGD becomes less beneficial, and momentum tends to outperform it.

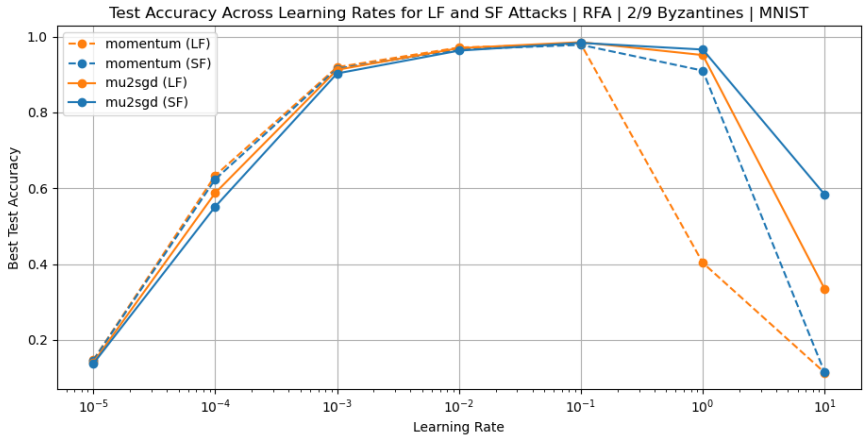


Figure 7. Performance comparison of μ^2-SGD with momentum across a wider range of learning rates under *label-flipping* and *sign-flipping* attacks with 2/9 Byzantine workers on the MNIST dataset (Conf. 3 in Table 2).

E.2.2. CIFAR-10

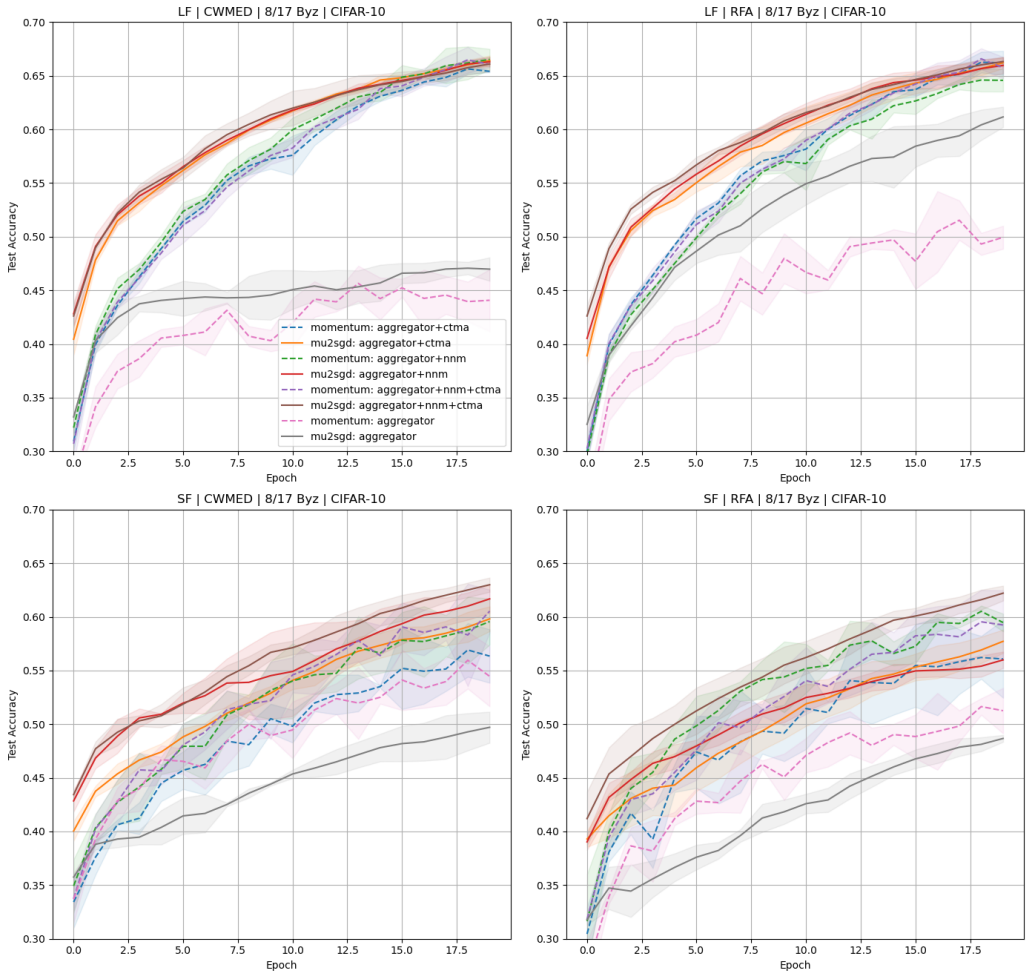


Figure 8. Performance comparison of CTMA with existing meta-aggregators, and μ^2-SGD with momentum under *sign-flipping* and *label-flipping* attacks with 8/17 Byzantine workers on the CIFAR-10 dataset (Conf. 2 in Table 2).

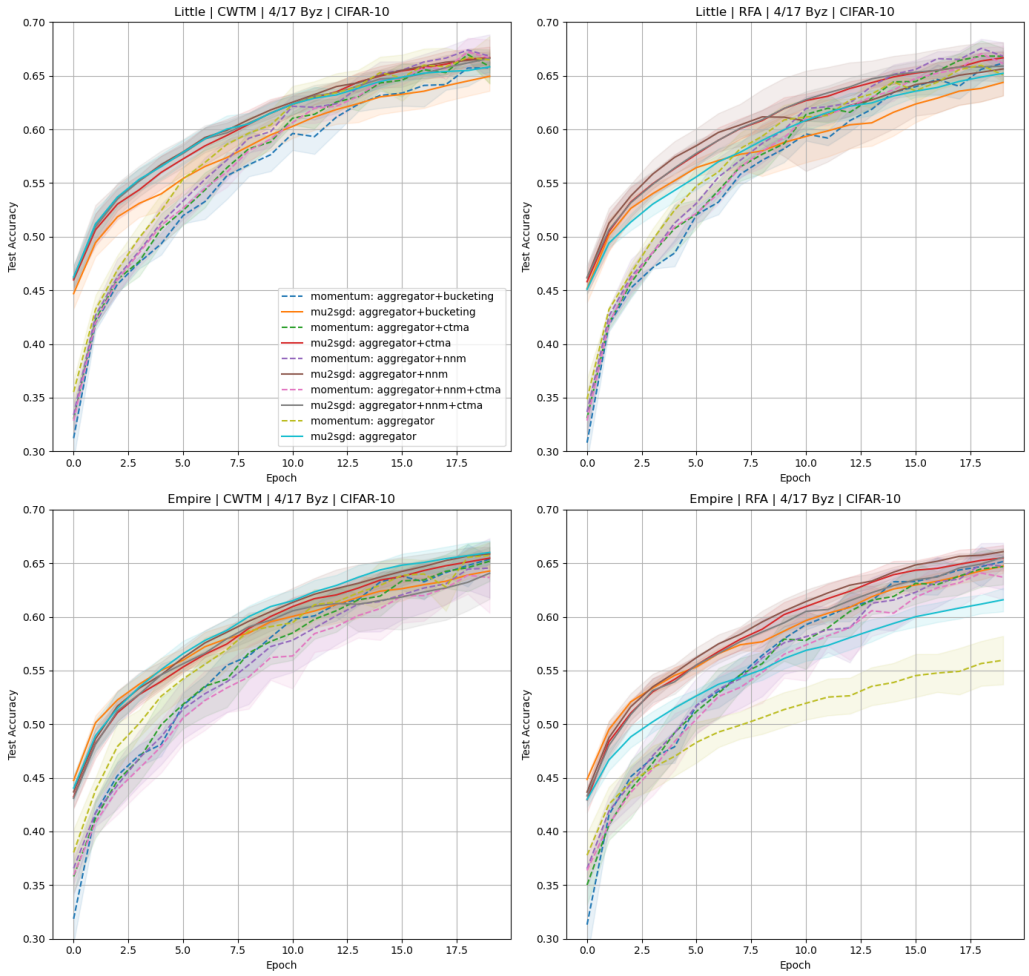


Figure 9. Performance comparison of CTMA with existing meta-aggregators, and μ^2-SGD with momentum under SOTA low-variance attacks, *empire* and *little*, with 4/17 Byzantine workers on the CIFAR-10 dataset (Conf. 2 in Table 2).

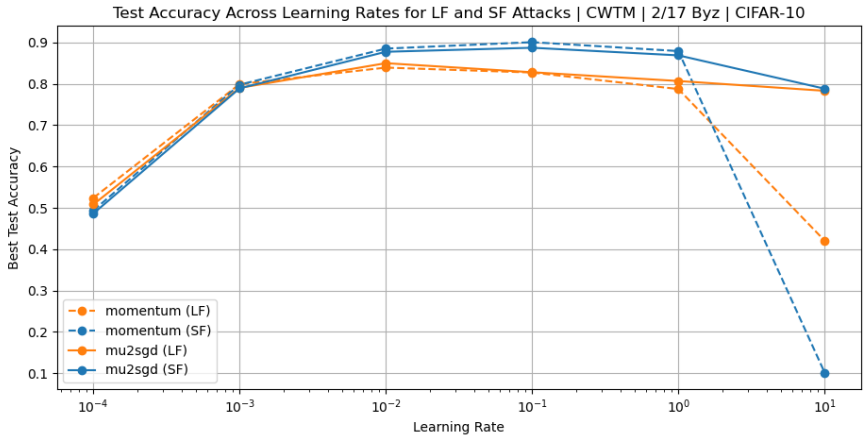


Figure 10. Performance comparison of μ^2-SGD with momentum across a wider range of learning rates under *label-flipping* and *sign-flipping* attacks with 2/17 Byzantine workers on the CIFAR-10 dataset (Conf. 4 in Table 2).