Moshpit SGD: Communication-Efficient Decentralized Training on Heterogeneous Unreliable Devices

Max Ryabinin[*]
Yandex, Russia
HSE University, Russia

Eduard Gorbunov[*]
MIPT, Russia
HSE University, Russia
Yandex, Russia

Vsevolod Plokhotnyuk
Yandex, Russia
HSE University, Russia

Gennady Pekhimenko
University of Toronto, Canada
Vector Institute, Canada

Abstract

Training deep neural networks on large datasets can often be accelerated by using multiple compute nodes. This approach, known as distributed training, can utilize hundreds of computers via specialized message-passing protocols such as Ring All-Reduce. However, running these protocols at scale requires reliable high-speed networking that is only available in dedicated clusters. In contrast, many real-world applications, such as federated learning and cloud-based distributed training, operate on unreliable devices with unstable network bandwidth. As a result, these applications are restricted to using parameter servers or gossip-based averaging protocols. In this work, we lift that restriction by proposing Moshpit All-Reduce — an iterative averaging protocol that exponentially converges to the global average. We demonstrate the efficiency of our protocol for distributed optimization with strong theoretical guarantees. The experiments show 1.3x speedup for ResNet-50 training on ImageNet compared to competitive gossip-based strategies and 1.5x speedup when training ALBERT-large on preemptible compute nodes.

1 Introduction

Many recent influential discoveries in deep learning were enabled by the trend of scaling model and dataset size. Over the last decade, computer vision has grown from training models with 60 million parameters [1] on 1.3 million images [2] to 15 times more parameters [3] and 200 times more training data [4]. In natural language processing, the state-of-the-art language models [5] with 175 billion parameters are trained on over 570GB of texts, and even this does not saturate the model quality [6]. Training these large models can take years even with a top-of-the-line GPU server [7]. As a result, researchers and practitioners often have to run distributed training with multiple machines [8].

The dominant approach to distributed deep learning is data-parallel training [9], where each worker processes a fraction of the training batch and then exchanges its gradients with peers. If done naïvely, the gradient exchange step can overload the network as the number of workers increases. To combat this issue, modern distributed training algorithms take advantage of communication-efficient protocols, such as all-reduce [10]. These protocols allow workers to collectively compute the global average gradient with a constant communication overhead, regardless of the total number of peers.

[*] Equal contribution. Correspondence to mryabinin0@gmail.com.

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

However, this efficiency makes the protocols more fragile: if any single participant fails or takes too long to process its batch, all other nodes are stalled. Therefore, scaling all-reduce protocols beyond a couple of servers requires specialized infrastructure with dedicated ultra-high bandwidth networking [8]. This kind of infrastructure is notoriously expensive compared to regular GPU servers or preemptible cloud VMs (see Appendix A for details).

Hence, it is tempting to consider distributed training on cheap unreliable instances as a cost-efficient alternative. A similar scenario arises in federated learning [11], where a single model is trained on heterogeneous devices due to privacy concerns. In both scenarios, workers use a shared network, where both latency and bandwidth can vary drastically due to interference from other users [12]. Furthermore, compute nodes are also subject to failure (or preemption) caused by factors beyond the protocol's control.

Running large-scale distributed training in these circumstances requires fault- and latency-tolerant algorithms [14, 15]. Most of these algorithms replace all-reduce averaging with **gossip**: each participant periodically downloads the latest parameters from their neighbors in a sparsely connected communication graph and averages the results. The updates gradually propagate through the graph over multiple rounds of averaging. However, the communication required to perform gossip grows linearly with the number of neighbors. Hence, when scaling to hundreds of peers, decentralized SGD has to keep the communication graph sparse, slowing down the convergence.

In this work, we propose an alternative approach. Instead of relying on a predefined communication graph, participants dynamically organize themselves into groups using a fully decentralized matchmaking algorithm called **Moshpit All-Reduce**. This strategy allows us to use communication-efficient all-reduce protocols that significantly reduce the network load compared to gossip-based averaging, while still being able to operate in unreliable hardware and network conditions.

Our contributions can be summarized as follows:

- We propose **Moshpit All-Reduce** — a novel decentralized averaging protocol for large-scale training with unreliable communication-constrained devices. According to our analysis, this method has exponential convergence rate independent of network topology and size.
- Armed with this averaging protocol, we develop **Moshpit SGD** for distributed optimization. We derive convergence rates for this algorithm and establish its equivalence to Centralized (Local) SGD in terms of iteration complexity under realistic assumptions.
- Our experiments demonstrate that Moshpit All-Reduce is significantly more efficient under network latency in realistic conditions. In particular, we train ResNet-50 on ImageNet to 75% accuracy 1.3 times faster than existing decentralized training algorithms and pretrain ALBERT-large 1.5 times faster on preemptible cloud VMs.[2]

2 Related Work

2.1 Data parallel training

The most popular way to accelerate neural network training with multiple devices is data-parallel training [9, 16, 17]. On each optimization step, this strategy splits the training batch among participants. Each participant then runs forward and backward passes to obtain gradients of the objective function on their part of the training batch. After that, we can aggregate the gradients from workers and perform an optimization step. There are two main strategies for this aggregation.

Historically, the first solution to gradient aggregation was to use Parameter Server (PS) [18]: a separate process or a dedicated server that keeps track of model parameters and optimizer statistics. After each round, the PS accumulates the gradients from each worker and updates the model parameters using SGD or any other optimizer, such as Adam [19]. Finally, the server distributes the updated model parameters to workers.

This strategy is robust and easy to implement, but it requires the server to regularly download full model gradients from every single worker. As a result, the parameter server can quickly become a bottleneck for large-scale training [20]. Since the original PS, researchers have proposed several modifications that reduce the communication load: accumulating multiple batches [22], compression [23, 24], server sharding [25, 26]. A more detailed overview is given in Appendix B.

[2] Implementation and code of experiments are at `github.com/yandex-research/moshpit-sgd`.

In turn, many practical distributed training systems have instead switched to averaging with All-Reduce [16, 27, 28, 17]. This name refers to a collection of protocols originally developed for HPC applications. Workers can follow these protocols to collectively compute the average[3] gradient more efficiently than with a central server.

2.2 Communication-efficient All-Reduce

There are several all-reduce protocols optimized for different network topologies. The simplest one is known as Butterfly All-Reduce [10]. Each of N participants splits its local vector into N chunks. Then, i-th worker aggregates i-th chunk of data from all peers and sends back the averaged chunk.

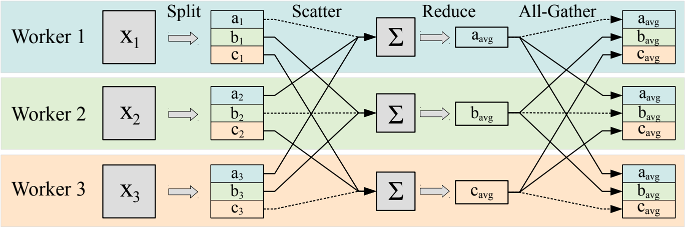

Figure 1: A schematic illustration of Butterfly All-Reduce.

As long as the vector size s is greater than N, this protocol uses $\mathcal{O}\left(s \times \frac{N-1}{N}\right)$ total bandwidth on each worker. However, it requires all-to-all communication, which is not always practical for the HPC infrastructure due to network contention [10]. As a result, real-world systems typically use Ring or Tree All-Reduce, where each worker only communicates with a small subset of its peers.

These protocols enable highly efficient and scalable averaging with $\mathcal{O}(1)$ or $\mathcal{O}(\log N)$ total communication per worker, but they also share a common drawback: they cannot tolerate node failures or network instability. If any single participant fails to execute its part or takes long to respond, this paralyzes all other workers.

2.3 Distributed training in unstable conditions

Some distributed training applications must deal with unstable network bandwidth and/or unreliable workers. This issue is most prevalent in federated learning [11, 29, 30]. When dealing with privacy-sensitive data distributed across multiple actors, such as hospital servers [31, 32] or mobile phones [33, 34], one must train the model using whichever hardware and network available to those actors.

Another important motivational factor is cost: HPC-grade infrastructure can be prohibitively expensive, pushing researchers and practitioners towards commodity servers or preemptible cloud VMs that are significantly cheaper (see Appendix A). Another solution is to use volunteer computing [35, 36] with abundant, but even less reliable, compute resources.

Training under these conditions requires specialized strategies. At a small scale, one can deploy one or a few reliable parameter servers to aggregate the updates from workers. This strategy can tolerate individual node failures [37], but scales poorly due to the reasons discussed in Section 2.1.

2.4 Decentralized training

If there are too many participants for PS, it can be advantageous to use decentralized SGD via **gossip-based** averaging [38, 39, 14]. In this scenario, participants form a sparse graph: each worker periodically downloads parameters from its neighbors and mixes them with local parameters.

In essence, gossip-based averaging removes the communication bottlenecks of PS at the cost of using different local parameters on each peer. That said, gossip-based optimization algorithms can match, and sometimes even outperform, their centralized counterparts in terms of training speed [40, 41, 42, 14, 43]. However, the convergence properties of gossip averaging and gossip-based optimization methods significantly depend on the communication graph through the spectral properties of the mixing matrix [44, 42] or the Laplacian matrix of the network [45, 46].

[3] All-Reduce works with any commutative associative operation, such as min, max, or product.

Consequently, as the number of peers increases, gossip-based averaging has to either increase the number of neighbors (hence more communication) or accept slower convergence speed. Because of this, gossip is less communication-efficient than all-reduce algorithms reviewed in Section 2.2. However, gossip-based algorithms are more robust to changes, which makes them applicable to time-varying networks [47, 48, 49, 50] and federated learning [51, 52, 53].

3 Moshpit SGD

Large-scale training with unreliable participants requires a protocol that is both communication-efficient and fault-tolerant. Unfortunately, existing methods have only provide one of these properties. To better address our conditions, we propose Moshpit All-Reduce — a fully decentralized averaging protocol that combines the efficiency of all-reduce and the fault tolerance of gossip-based averaging.

The rest of this section is organized as follows:

- Section 3.1 describes the protocol and proves its correctness and communication efficiency;
- Section 3.2 provides the analysis of the protocol and proves exponential convergence rate for averaging and the rate matching the one of centralized Local-SGD for optimization;
- Section 3.3 contains implementation details for training with heterogeneous compute nodes.

3.1 Moshpit All-Reduce

The core idea of Moshpit All-Reduce is that workers perform averaging in small independent groups. That way, a single failed participant would only affect his current group. In turn, the composition of each group should be chosen dynamically to converge in the least number of steps. Ideally, if there are 9 peers with local parameters θ, we can average them in 2 rounds, as demonstrated in Figure 2:

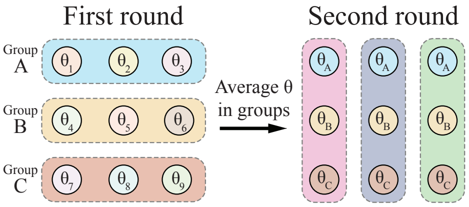

Figure 2: Example averaging order for 9 peers in 2 rounds. On each round, peers are split into 3 groups that run All-Reduce in parallel.

Algorithm 1 Moshpit All-Reduce (for i-th peer)

```
Input: parameters $\{\theta_j\}_{j=1}^N$, number of peers $N$, $d$, $M$, number of iterations $T$, peer index $i$
$\theta_i^0 := \theta_i$
$C_i^0 :=$ get_initial_index(i)
for $t \in 1 \dots T$ do
    DHT$[C_i^{t-1}, t]$.add(address$_i$)
    Matchmaking() // wait for peers to assemble
    peers$_t :=$ DHT.get($[C_i^{t-1}, t]$)
    $\theta_i^t, c_i^t :=$ AllReduce($\theta_i^{t-1}$, peers$_t$)
    $C_i^t := (C_i^{t-1}$[1:]$, c_i^t)$ // same as eq. (1)
end for
Return $\theta_i^T$
```

To achieve this in a decentralized system, we use Distributed Hash Tables (DHT) — a decentralized key-value storage; Appendix B contains its more detailed description. On each averaging round:

- Each worker computes its group key C_i;
- Workers add their network addresses to the DHT key corresponding to C_i;
- Each worker can now fetch a full list of peers that have the same C_i and run All-Reduce with those peers.

Unfortunately, the averaging structure from Figure 2 is impossible to maintain when participants are constantly joining, leaving, and failing. However, we can achieve equivalent results without global structure using a simple rule: *if two peers were in the same group in round t, they must choose different groups in round* $t+1$.

A natural way to enforce this rule is to take advantage of the chunk indices from Butterfly All-Reduce (see Figure 1). Recall that each worker accumulates a *unique* chunk of parameters defined by an index c_i. By setting $C_i := c_i$, we can guarantee that any workers that were in the same group at a round t will have different group indices in round $t+1$.

This averaging scheme can be generalized to more than two dimensions in order to fit a larger number of peers or reduce the group size. For a d-dimensional hypercube, nodes should find groups of peers that they have not communicated with during $d{-}1$ previous rounds. To that end, we define C_i as tuples containing chunk indices from $d{-}1$ previous rounds (t denotes the communication round):

$$C_i^t := (c_i^{t-d+1}, c_i^{t-d+2}, \ldots, c_i^t). \tag{1}$$

The above intuition can be formalized with Algorithm 1. Here, N peers form a virtual d-dimensional grid with M peers per row and average their parameters θ_i over T rounds. $\texttt{DHT}[\cdot]$ is a shortcut for using the DHT to add or retrieve values for a given key. The `Matchmaking` step corresponds to the decentralized matchmaking procedure that organizes active workers with the same index into groups, described in detail in Appendix E. In turn, `AllReduce` denotes running all-reduce to compute the average θ in a given group. The `get_initial_index` function takes the peer index i and returns $d{-}1$ integers in range $[0, M)$ such that the size of initial groups does not exceed M. This way, the groups formed on subsequent rounds will also have at most M participants. One possible strategy is:

$$\texttt{get\_initial\_index}(i) = \left(\lfloor i/M^{d-1} \rfloor \bmod M\right)_{j \in \{1,\,\ldots,\,d\}} \tag{2}$$

If $N{=}M^d$ and there are no node/network failures, Algorithm 1 is equivalent to Torus All-Reduce [54], achieving the exact average after d rounds of communication (see Appendix C.1). However, our typical use case is far from this perfect scenario; for example, some groups can have less than M members. Furthermore, a peer might fail during all-reduce, causing its groupmates to skip a round of averaging. Still, Moshpit All-Reduce is applicable even in these conditions:

Theorem 3.1 (Correctness)**.** *If all workers have a non-zero probability of successfully running a communication round and the order of $\texttt{peers}_t$ is random, then all local vectors θ_i^t converge to the global average with probability 1:*

$$\forall i, \left|\left|\theta_i^t - \frac{1}{N}\sum_i \theta_i^0\right|\right|_2^2 \xrightarrow[t\to\infty]{} 0. \tag{3}$$

Proof (sketch, complete in Appendix C.2). Running all-reduce with a subset of peers preserves the invariant $\frac{1}{N}\sum_i \theta_i^t = \frac{1}{N}\sum_i \theta_i^{t-1}$ and reduces the deviation of θ_i^t from the overall average. □

Complexity. The matchmaking protocol is implemented over Kademlia DHT [55], meaning that each read and write operation needs at most $\mathcal{O}(\log N)$ requests and $\mathcal{O}(M)$ bandwidth to load $\texttt{peers}_t$.

After the matchmaking is over, each group runs a single all-reduce round to compute the average. In principle, Moshpit Averaging can use any general-purpose all-reduce protocol. We opted for a butterfly-like version (Figure 1), as it is simpler than Ring All-Reduce while still being communication-efficient. The communication complexity of this algorithm is $\mathcal{O}\left(\max(s, M) \times \frac{M-1}{M}\right)$, where s is the size of vector θ. Thus, the total time complexity of Algorithm 1 becomes:

$$\mathcal{O}\left(T \times \left[\log_2 N + M + \max(s, M) \times \frac{M-1}{M}\right]\right). \tag{4}$$

This compares favorably to gossip, where network load grows linearly with the number of neighbors.

3.2 Convergence analysis

3.2.1 Mixing properties of Moshpit Averaging

As stated in the previous section, Moshpit All-Reduce computes the exact average when $N = M^d$, which cannot be guaranteed in practice. Therefore, additional analysis is needed to establish how quickly Moshpit Averaging approximates the actual average of N vectors stored on peers.

In the following theorem, we provide such analysis for a simplified version of Moshpit Averaging. One can find the full proof in Appendix C.3.

Theorem 3.2. *Consider a modification of Moshpit All-Reduce that works as follows: at each iteration $k \geq 1$, 1) peers are randomly split in r disjoint groups of sizes $M_1^k, \ldots, M_r^k$ in such a way that $\sum_{i=1}^r M_i^k = N$ and $M_i^k \geq 1$ for all $i = 1, \ldots, r$ and 2) peers from each group compute their group average via All-Reduce. Let $\theta_1, \ldots, \theta_N$ be the input vectors of this procedure and $\theta_1^T, \ldots, \theta_N^T$ be the outputs after T iterations. Also, let $\overline{\theta} = \frac{1}{N}\sum_{i=1}^N \theta_i$ Then,*

$$\mathbb{E}\left[\frac{1}{N}\sum_{i=1}^N \|\theta_i^T - \overline{\theta}\|^2\right] = \left(\frac{r-1}{N} + \frac{r}{N^2}\right)^T \frac{1}{N}\sum_{i=1}^N \|\theta_i - \overline{\theta}\|^2. \tag{5}$$

Algorithm 2 Moshpit SGD

```
Input: starting point $\theta^0$, learning rate $\gamma > 0$, communication period $\tau \geq 1$
for $k = 0, 1, \ldots$ do
    for each peer $i \in P_{k+1}$ in parallel do
        Compute the stochastic gradient $g_i^k$ at the current point $\theta_i^k$
        if $k + 1 \mod \tau = 0$ then
            $\theta_i^{k+1} = \text{Moshpit All-Reduce}_{j \in P_{k+1}}(\theta_j^k - \gamma g_j^k)$ for $i$-th peer (Algorithm 1)
        else
            $\theta_i^{k+1} = \theta_i^k - \gamma g_i^k$
        end if
    end for
end for
```

In particular, this result implies that even if workers are randomly split into pairs at each iteration, the simplified version of Moshpit Averaging makes the average distortion (the left-hand side of Equation 5) less than ε in expectation after $\mathcal{O}\left(\log(1/\varepsilon)\right)$ iterations. That is, this algorithm finds ε-accurate average on each node with the rate that *does not* depend on the spectral properties of the communication graph like gossip and its variants (see Section 2.4 and Appendix B.1). Since Moshpit Averaging prevents two peers from participating in the same groups during successive iterations, the actual algorithm should find ε-accurate averages on participating peers even faster than Equation 5 predicts. Moreover, in Appendix C.3 we explain how this result can be generalized to the case when $\{M_i^k\}_{i=1}^N$ and r depends on k or even is random. In Appendix C.4, we also provide the guarantees measuring how fast Algorithm 1 reduces the variance when averaging random vectors.

3.2.2 Moshpit SGD

We consider a classical distributed optimization problem

$$\min_{\theta \in \mathbb{R}^n} \left\{ f(\theta) = \frac{1}{N} \sum_{i=1}^{N} f_i(\theta) \right\}, \tag{6}$$

where N is the number of workers and worker i has access only to the function f_i.

We propose a new algorithm called Moshpit SGD to solve this problem (see Algorithm 2). In this algorithm, workers perform independent local SGD steps and periodically synchronize their parameters θ_i^k with other peers using Moshpit All-Reduce. Moreover, we define the indices of participating nodes at iteration k as P_{k+1} ($P_0 = \{1, \ldots, N\}$) allowing peers to vanish.

First of all, we list the key assumptions that we use in the convergence analysis of Moshpit SGD.

Assumption 3.1 (Bounded variance)**.** *We assume that for all $k \geq 0$ and $i = 1, \ldots, N$ stochastic gradients g_i^k satisfy $\mathbb{E}\left[g_i^k \mid \theta_i^k\right] = \nabla f_i(\theta_i^k)$ and*

$$\mathbb{E}\left[\|g_i^k - \nabla f_i(\theta_i^k)\|^2 \mid \theta_i^k\right] \quad \leq \quad \sigma^2. \tag{7}$$

This assumption is classical in the stochastic optimization literature [56, 57]. We notice that our analysis can be generalized to the settings when the stochastic gradients satisfy less restrictive assumptions such as expected smoothness [58] or have more sophisticated structure similar to [59] using the theoretical framework from [60].

The following assumption controls the averaging properties and the effect of the peers' vanishing.

Assumption 3.2 (Averaging quality & peers' vanishing)**.** *We assume that the vanishing of peers does not change the global average of the iterates of Moshpit SGD too much, i.e., $P_{k+1} \subseteq P_k$ and $|P_k| \geq N_{\min}$ for all $k \geq 0$, $|P_{a\tau}| \leq 2|P_{a(\tau+1)}|$ for all non-negative integers $a \geq 0$, and there exist such $\widetilde{\theta} \in \mathbb{R}^n$ and a sequence of non-negative numbers $\{\Delta_{pv}^k\}_{k \geq 0}$ that $\forall k \geq 0$*

$$\mathbb{E}\left[\langle \theta^{k+1} - \widehat{\theta}^{k+1}, \theta^{k+1} + \widehat{\theta}^{k+1} - 2\widetilde{\theta} \rangle\right] \leq \Delta_{pv}^k, f \text{ convex;} \tag{8}$$

$$\mathbb{E}\Big[\langle \nabla f(\theta^k), \theta^{k+1} - \widehat{\theta}^{k+1} \rangle + L\|\widehat{\theta}^{k+1} - \theta^{k+1}\|^2\Big] \leq \Delta_{pv}^k, f \text{ non-convex, } L\text{-smooth, (Def. D.1)} \tag{9}$$

where $N_k = |P_k|$, $\theta^{k+1} = \frac{1}{N_{k+1}} \sum_{i \in P_{k+1}} \theta_i^{k+1}$, and $\widehat{\theta}^{k+1} = \frac{1}{N_k} \sum_{i \in P_k} (\theta_i^k - \gamma g_i^k)$ for $k \geq 0$.

Moreover, we assume that for some $\delta_{aq} \geq 0$ and for all non-negative integers $a \geq 0$,

$$\mathbb{E}\left[\frac{1}{N_{a\tau}}\sum_{i\in P_{a\tau}}\|\theta_i^{a\tau} - \theta^{a\tau}\|^2\right] \quad \leq \quad \gamma^2\delta_{aq}^2. \tag{10}$$

If $P_k = P_{k+1} = \{1, \ldots, N\}$ for all $k \geq 0$, i.e., peers do not vanish, then $\theta^k = \widehat{\theta}^k$ and properties (8, 9) hold with $\Delta_{pv}^k \equiv 0$ for all $k \geq 0$. Moreover, according to the mixing properties of Moshpit Averaging established in Theorem 3.2, inequality 10 holds after $\mathcal{O}\left(\log\left({}^1/_{\gamma^2\delta_{aq}^2}\right)\right)$ iterations of Algorithm 1. Therefore, the assumption above is natural and well-motivated.

Under these assumptions, we derive the convergence rates both for convex and non-convex problems. The full statements and complete proofs are deferred to Appendix D.

Theorem 3.3 (Convex case). *Let $f_1 = \ldots = f_N = f$, function f be μ-strongly convex (Def. D.2) and L-smooth (see Def. D.1), and Assumptions 3.1 and 3.2 hold with $\Delta_{pv}^k = \delta_{pv,1}\gamma\mu\mathbb{E}[\|\theta^k - \theta^*\|^2] + \gamma^2\delta_{pv,2}^2$ and $\widetilde{\theta} = \theta^*$, where $\theta^* \in \operatorname{argmin}_{\theta\in\mathbb{R}^n} f(\theta)$ and $\delta_{pv,1} \in [0, 1)$, $\delta_{pv,2} \geq 0$. Then there exists a choice of γ such that $\mathbb{E}\left[f(\overline{\theta}^K) - f(\theta^*)\right] \leq \varepsilon$ after K iterations of Moshpit SGD, where K equals*

$$\widetilde{\mathcal{O}}\left(\frac{L}{(1-\delta_{pv,1})\mu} + \frac{\delta_{pv,2}^2 + {}^{\sigma^2}/_{N_{\min}}}{(1-\delta_{pv,1})\mu\varepsilon} + \sqrt{\frac{L((\tau-1)\sigma^2 + \delta_{aq}^2)}{(1-\delta_{pv,1})^2\mu^2\varepsilon}}\right),\ \mu > 0; \tag{11}$$

$$\mathcal{O}\left(\frac{LR_0^2}{\varepsilon} + \frac{R_0^2(\delta_{pv,2}^2 + {}^{\sigma^2}/_{N_{\min}})}{\varepsilon^2} + \frac{R_0^2\sqrt{L((\tau-1)\sigma^2 + \delta_{aq}^2)}}{\varepsilon^{3/2}}\right),\ \mu = 0, \tag{12}$$

where $\overline{\theta}^K = \frac{1}{W_K}\sum\limits_{k=0}^{K}\frac{1}{N_k}\sum\limits_{i\in P_k} w_k\theta_i^k$, $w_k = (1-\gamma\mu)^{-(k+1)}$, $W_K = \sum_{k=0}^K w_k$, $R_0 = \|\theta^0 - \theta^*\|$ and $\widetilde{\mathcal{O}}(\cdot)$ hides constant and $\log({}^1/_\varepsilon)$ factors.

That is, if $\delta_{pv,1} \leq {}^1/_2$, $N_{\min} = \Omega(N)$, $\delta_{pv,2}^2 = \mathcal{O}({}^{\sigma^2}/_{N_{\min}})$, and $\delta_{aq}^2 = \mathcal{O}((\tau-1)\sigma^2)$, then Moshpit SGD has the same iteration complexity as Local-SGD in the homogeneous case [61, 62]. However, the averaging steps of Moshpit SGD are much faster than those of the parameter-server architecture when the number of peers is large. Also, unlike the state-of-the-art convergence guarantees for Decentralized Local-SGD [63], our bounds do not depend on the spectral properties of the communication graph (see Appendix B.1 for the details).

Theorem 3.4 (Non-convex case). *Let $f_1 = \ldots = f_N = f$, function f be L-smooth and bounded from below by f_*, and Assumptions 3.1 and 3.2 hold with $\Delta_{pv}^k = \delta_{pv,1}\gamma\mathbb{E}[\|\nabla f(\theta^k)\|^2] + L\gamma^2\delta_{pv,2}^2$, $\delta_{pv,1} \in [0, {}^1/_2)$, $\delta_{pv,2} \geq 0$. Then there exists such choice of γ that $\mathbb{E}\left[\|\nabla f(\theta_{rand}^K)\|^2\right] \leq \varepsilon^2$ after K iterations of Moshpit SGD, where K equals*

$$\mathcal{O}\left(\tfrac{L\Delta_0}{(1-2\delta_{pv,1})^2\varepsilon^2}\left[1 + \tau\sqrt{1-2\delta_{pv,1}} + \tfrac{\delta_{pv,2}^2 + {}^{\sigma^2}/_{N_{\min}}}{\varepsilon^2} + \tfrac{\sqrt{(1-2\delta_{pv,1})(\delta_{aq}^2 + (\tau-1)\sigma^2)}}{\varepsilon}\right]\right),$$

$\Delta_0 = f(\theta^0) - f(\theta^*)$ and θ_{rand}^K is chosen uniformly from $\{\theta^0, \theta^1, \ldots, \theta^{K-1}\}$ defined in As. 3.2.

Again, if $\delta_{pv,1} \leq {}^1/_3$, $N_{\min} = \Omega(N)$, $\delta_{pv,2}^2 = \mathcal{O}({}^{\sigma^2}/_{N_{\min}})$, and $\delta_{aq}^2 = \mathcal{O}((\tau-1)\sigma^2)$, then the above theorem recovers the state-of-the-art results in the non-convex case for Local-SGD [64, 63].

3.3 Implementation details

Training on heterogeneous unreliable hardware also poses a number of engineering challenges. The most obvious one is that the system must be able to recover from node failures. To address this challenge, we use a fully decentralized infrastructure where all information is replicated in a Distributed Hash Table; see Appendix B.5 for details. When a new worker joins midway through training, it can download the latest model parameters and metadata from any other peer (see Appendix F). Another challenge arises when devices in a group have uneven network bandwidth. In that case, we dynamically adjust the communication load of each peer to avoid being bottlenecked. More information on this procedure can be found in Appendix G.

4 Experiments

In this section, we conduct empirical evaluation of the proposed averaging protocol and its corresponding optimization algorithm. First, we check the theoretical properties of Moshpit All-Reduce in a controlled setup (Section 4.1). Then, we compare Moshpit SGD with other distributed methods on practical tasks of image classification and masked language model pretraining (Sections 4.2 and 4.3).

4.1 Decentralized averaging

In this series of experiments, we aim to empirically verify the convergence and fault tolerance properties proven in Section 3.2. To measure this in a controlled setting, we create peers with parameters that are scalar values drawn from the standard Gaussian distribution. We study the convergence of different distributed methods with respect to the number of workers N and their individual failure rate for a single iteration of averaging p (failed peers return in the next round).

We compare Moshpit Averaging with the following algorithms from prior work: All-Reduce (with restarts in case of node failures), Gossip, PushSum (equivalent to the method described in [15]). Also, we provide the results of averaging in random groups as a simpler version of our approach. However, the implementation of group averaging maintains approximately the same group size across all iterations: this property might be hard to achieve in a decentralized setting, and as a result, the estimate of this method's performance should be considered highly optimistic.

We report the average squared difference between the worker parameters and the actual average of all values; the results are averaged across 100 restarts from different random initializations. We compare the convergence for 512–1024 peers and consider failure probabilities ranging from 0 to 0.01. For Moshpit Averaging and random group averaging, we use groups of size 32, which corresponds to $M = 32$ and $d = 2$ for Algorithm 1.

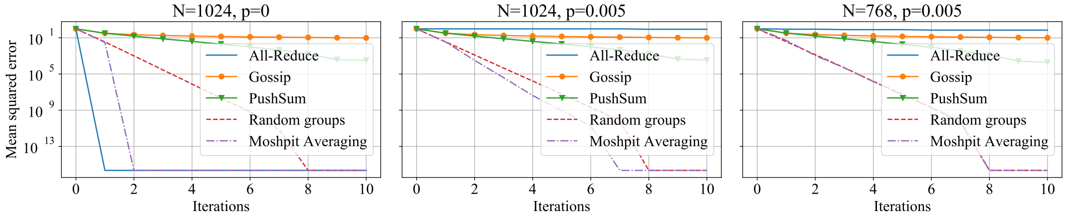

Figure 3: Convergence of averaging algorithms in different configurations.

Figure 3 displays the results of experiments for several combinations of N and p; the complete results with additional grid configurations are available in Appendix I. We make several key observations:

1. When the failure rate of each peer is zero, standard All-Reduce predictably computes the average faster than all other methods. However, as soon as p reaches a value of at least 0.005, the number of retries needed for the success becomes prohibitively high.

2. Previous decentralized averaging methods, such as Gossip or PushSum, require significantly more iterations for convergence to the global average than Moshpit All-Reduce, likely due to the structure of their communication graphs.

3. As discussed in Section 3.1, when the total number of peers is equal to the grid capacity and there are no failures, Moshpit All-Reduce matches the result of regular All-Reduce with the number of steps equal to the number of grid dimensions (2 in this case).

4. Averaging in random groups can perform comparably to Moshpit Averaging when the number of peers is less than half of the grid capacity. The reason for this behavior is that when the workers do not fully occupy the grid, the group sizes are no longer guaranteed to be equal across groups and across iterations. In the worst case, there can be groups of only one peer for certain grid coordinates, which may significantly affect the convergence. However, as the grid utilization grows, Moshpit Averaging starts to outperform random group averaging. Moreover, even if we use 512 peers, arranging them in a proper 8x8x8 grid leads to faster convergence.

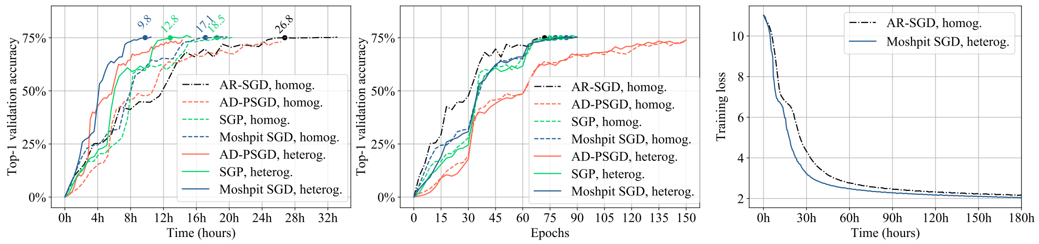

Figure 4: **(Left, Middle)** ResNet-50 top-1 validation accuracy for ImageNet as a function of training time (left) and epochs (middle). **(Right)** Full training objective (MLM + SOP) of ALBERT-large on BookCorpus as a function of training time.

4.2 ImageNet training

Here, we evaluate the performance of Moshpit SGD in distributed training. More specifically, we train ResNet-50 [65] on the ILSVRC [2] dataset, following the training protocol of [16]. Trainers use SGD with Nesterov momentum with a batch size of 256 and 32-bit precision regardless of the GPU type[4]. We evaluate the following training strategies:

- **All-Reduce SGD (AR-SGD)** — traditional distributed training with all-reduce gradient averaging;
- **Asynchronous Decentralized Parallel SGD (AD-PSGD)** — parallel SGD that runs gossip communication in a cycle: each worker averages parameters with 2 neighbors [66]. Communication rounds are overlapped with computation;
- **Stochastic Gradient Push (SGP)** — a more advanced algorithm with an exponential communication graph and push-based communication [15];
- **Moshpit SGD** — similar to **SGP**, but with 1 round of Moshpit Averaging instead of PushSum.

We report top-1 validation accuracy as a function of training time in two experimental setups:

- **Homogeneous**: 16 servers with a single Tesla V100-PCIe GPU, 6 CPU cores, and 64GB RAM.
- **Heterogeneous**: a total of 81 GPUs (V100, 1080Ti, and P40) across 64 servers and workstations.[5]

All servers and workstations communicate over the network with 1Gb/s Ethernet (non-dedicated symmetric bandwidth). The machines are located in two data centers and one office within 300 km of one another. The communication latency is 1–6ms depending on the location. To simulate shared usage, at the beginning of each communication round we inject additional latency sampled from the exponential distribution [67] with the mean of 100ms.

For Moshpit SGD, we use a two-dimensional "grid" with 4 and 8 groups for homogeneous and heterogeneous setups respectively. For AD-PSGD, we attempt to compensate for slow convergence by training for 60 more epochs without changing the learning rate schedule. Finally, we only report AR-SGD in the first setup, as it is unsuitable for heterogeneous hardware.

The results in Figure 4 (Left) demonstrate that the two most efficient strategies for our setting are Moshpit SGD and SGP. In the **homogeneous** setup, Moshpit is only slightly more efficient than SGP, likely due to higher efficiency of all-reduce. This advantage increases to over 30% for the **heterogeneous** setup with 64 servers. In turn, AR-SGD demonstrates the best performance per iteration, but its training time is by far the longest due to network latency (1.5× of Moshpit SGD). Finally, AD-PSGD predictably shows the best throughput (time per epoch), but achieves lower accuracy even after training for 150 epochs. We report results for smaller setups in Appendix J.

4.3 Masked Language Model training

Finally, we evaluate Moshpit All-Reduce training performance in the wild with preemptible cloud instances. For this experiment, we perform one of the most resource-demanding tasks in modern deep learning — unsupervised pretraining of Transformers [68, 69, 70, 5]. We opt for the ALBERT model [71] to make better use of communication-constrained devices. This model has fewer trainable parameters due to layer-wise weight sharing.

[4] For GPUs that cannot fit this into memory, we accumulate gradients over 2 batches of 128 examples.

[5] We provide a detailed configuration in Appendix H.

Specifically, we train ALBERT-large (18M parameters) on the BookCorpus [72] dataset, following the training setup from the original paper. We minimize the masked language modeling loss (MLM) along with the sentence order prediction loss (SOP) using the LAMB optimizer [17] with a global batch size of 4096 and sequence length 512. We measure convergence in terms of full training loss [73, 74]. Similarly to Section 4.2, we use two training setups:

- **Homogeneous:** a single cloud instance with 8 Tesla V100-PCIe GPUs and 56 vCPUs;
- **Heterogeneous:** a total of 66 preemptible GPUs, 32 of which are cloud T4, and the remaining 34 are various devices rented on a public marketplace.

Despite the fact that the latter setup has almost $3\times$ more raw compute[6], its hourly rent costs less than the homogeneous setup due to relying on preemptible instances[7]. This instance type is much cheaper than regular cloud instances, but it can be interrupted at any time. As a side-effect, the participants in **heterogeneous** setup are also spread across 3 continents with uneven network bandwidth, ranging from 100Mb/s to 1500Mb/s per worker. These limitations make it impractical to deploy conventional all-reduce protocols. By contrast, the fully decentralized nature of Moshpit SGD allows it to operate on unreliable nodes.

In this setup, the participants accumulate gradients over multiple local batches and use DHT to track the global batch size. Once the swarm collectively accumulates gradients over 4096 training samples, it runs 2 rounds of Moshpit All-Reduce with $M{=}8$ and $d{=}2$. Unfortunately, training with simple parameter averaging does not converge, likely due to diverging LAMB statistics. To mitigate this issue, workers recover "pseudo-gradients" [76, 77] after averaging to update the optimizer statistics.

Figure 4 (right) demonstrates that Moshpit SGD with a fully preemptible fleet of machines trains 1.5 times faster than the traditional data-parallel setup. The final loss achieved by two training strategies is the same within the margin of error. A closer investigation reveals that this speedup is entirely explained by the reduced iteration time. An interesting observation is that the iteration time of Moshpit SGD varies between 10–22 seconds, while AR-SGD consistently spends 25s per step. This can be explained by natural variation in the preemptible fleet size: there were 30–66 active participants depending on the resource availability.

5 Conclusion and future work

In this work, we propose Moshpit All-Reduce, a decentralized averaging protocol intended for distributed optimization in unstable and network-constrained environments. It has favorable theoretical properties when compared to gossip-based approaches and achieves considerable speedups in distributed training for image classification and masked language modeling.

Our approach was primarily designed for cloud-based training and federated learning, as well as for distributed training on unreliable instances; future work might explore additional settings, such as collaborative training of neural networks. Another potential research direction is to study the interactions of Moshpit All-Reduce with other methods that improve communication efficiency of distributed optimization, such as gradient compression. Finally, the idea of arranging All-Reduce nodes into groups can be improved to address specific issues that may arise in practice, such as the varying number of workers and their geographical distribution.

Acknowledgements

We would like to thank Anastasia Koloskova, Liudmila Prokhorenkova and Anton Osokin for helpful feedback and discussions. We are also grateful to the anonymous reviewers for their suggestions on improving the paper. Finally, we would like to thank Dmitry Afanasiev, Vladimir Aliev, Anand Jayarajan and Michael Solotky for their suggestions on the technical aspects of our study. This project was supported in part by the Canada Foundation for Innovation JELF grant, NSERC Discovery grant, AWS Machine Learning Research Award, and Facebook Faculty Research Award. The paper was also partially supported by by a grant for research centers in the field of artificial intelligence, provided by the Analytical Center for the Government of the Russian Federation in accordance with the subsidy agreement (agreement identifier 000000D730321P5Q0002) and the agreement with the Moscow Institute of Physics and Technology dated November 1, 2021 No. 70-2021-00138. The computational resources for the experiments were provided by the Amazon Research Awards program and Yandex.

[6] Based on official performance benchmarks [75].

[7] Please refer to Appendix H for full experimental setups.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[3] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, S. Gelly, and N. Houlsby. Big transfer (bit): General visual representation learning. In *ECCV*, 2020.

[4] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*, 2017.

[5] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[6] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.

[7] Chuan Li. Demystifying gpt-3 language model: A technical overview, 2020. "`https://lambdalabs.com/blog/demystifying-gpt-3`".

[8] Peter Mattson, Christine Cheng, Cody Coleman, Greg Diamos, Paulius Micikevicius, David Patterson, Hanlin Tang, Gu-Yeon Wei, Peter Bailis, Victor Bittorf, David Brooks, Dehao Chen, Debojyoti Dutta, Udit Gupta, Kim Hazelwood, Andrew Hock, Xinyuan Huang, Bill Jia, Daniel Kang, David Kanter, Naveen Kumar, Jeffery Liao, Guokai Ma, Deepak Narayanan, Tayo Oguntebi, Gennady Pekhimenko, Lillian Pentecost, Vijay Janapa Reddi, Taylor Robie, Tom St. John, Carole-Jean Wu, Lingjie Xu, Cliff Young, and Matei Zaharia. MLPerf Training Benchmark. In *Proceedings of the 3rd Conference on Machine Learning and Systems (MLSys'20)*, 2020.

[9] Leslie G Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33(8):103–111, 1990.

[10] Pitch Patarasuk and Xin Yuan. Bandwidth optimal all-reduce algorithms for clusters of workstations. *J. Parallel Distrib. Comput.*, 69(2):117–124, February 2009.

[11] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282, 2017.

[12] V. Persico, P. Marchetta, A. Botta, and A. Pescape. On network throughput variability in microsoft azure cloud. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2015.

[13] Valerio Persico, Pietro Marchetta, Alessio Botta, and Antonio Pescapè. Measuring network throughput in the cloud: The case of amazon ec2. *Computer Networks*, 93:408 – 422, 2015. Cloud Networking and Communications II.

[14] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 5330–5340, 2017.

[15] Mahmoud Assran, Nicolas Loizou, Nicolas Ballas, and Mike Rabbat. Stochastic gradient push for distributed deep learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 344–353. PMLR, 09–15 Jun 2019.

[16] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour, 2017.

[17] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations*, 2020.

[18] Mu Li. Scaling distributed machine learning with the parameter server. In *Proceedings of the 2014 International Conference on Big Data Science and Computing*, BigDataScience '14, New York, NY, USA, 2014. Association for Computing Machinery.

[19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

[20] Salem Alqahtani and Murat Demirbas. Performance analysis and comparison of distributed machine learning systems, 2019.

[21] Joost Verbraeken, Matthijs Wolting, Jonathan Katzy, Jeroen Kloppenburg, Tim Verbelen, and Jan S. Rellermeyer. A survey on distributed machine learning. *ACM Comput. Surv.*, 53(2), March 2020.

[22] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex Smola. Parallelized stochastic gradient descent. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23, pages 2595–2603. Curran Associates, Inc., 2010.

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

[24] Anastasia Koloskova, Sebastian Stich, and Martin Jaggi. Decentralized stochastic optimization and gossip algorithms with compressed communication. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3478–3487. PMLR, 09–15 Jun 2019.

[25] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc' aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, Quoc Le, and Andrew Ng. Large scale distributed deep networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25, pages 1223–1231. Curran Associates, Inc., 2012.

[26] Yimin Jiang, Yibo Zhu, Chang Lan, Bairen Yi, Yong Cui, and Chuanxiong Guo. A unified architecture for accelerating distributed DNN training in heterogeneous gpu/cpu clusters. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 463–479. USENIX Association, November 2020.

[27] Hiroaki Mikami, Hisahiro Suganuma, Pongsakorn U-chupala, Yoshiki Tanaka, and Yuichi Kageyama. Massively distributed sgd: Imagenet/resnet-50 training in a flash, 2019.

[28] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using gpu model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

[29] Aaron Segal, Antonio Marcedone, Benjamin Kreuter, Daniel Ramage, H. Brendan McMahan, Karn Seth, K. A. Bonawitz, Sarvar Patel, and Vladimir Ivanov. Practical secure aggregation for privacy-preserving machine learning. In *CCS*, 2017.

[30] K. A. Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé M Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards federated learning at scale: System design. In *SysML 2019*, 2019. To appear.

[31] Micah J. Sheller, Brandon Edwards, G. Anthony Reina, Jason Martin, Sarthak Pati, Aikaterini Kotrotsou, Mikhail Milchenko, Weilin Xu, Daniel Marcus, Rivka R. Colen, and Spyridon Bakas. Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. *Scientific Reports*, 10(1):12598, Jul 2020.

[32] Wenqi Li, Fausto Milletarì, Daguang Xu, Nicola Rieke, Jonny Hancox, Wentao Zhu, Maximilian Baust, Yan Cheng, Sébastien Ourselin, M. Jorge Cardoso, and Andrew Feng. *Privacy-Preserving Federated Brain Tumour Segmentation*, pages 133–141. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). SPRINGER, January 2019. 10th International Workshop on Machine Learning in Medical Imaging, MLMI 2019 held in conjunction with the 22nd International Conference on Medical Image Computing and Computer-Assisted Intervention, MICCAI 2019 ; Conference date: 13-10-2019 Through 13-10-2019.

[33] Andrew Hard, Chloé M Kiddon, Daniel Ramage, Francoise Beaufays, Hubert Eichner, Kanishka Rao, Rajiv Mathews, and Sean Augenstein. Federated learning for mobile keyboard prediction, 2018.

[34] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. Applied federated learning: Improving google keyboard query suggestions, 2018.

[35] Ekasit Kijsipongse, Apivadee Piyatumrong, and Suriya U-ruekolan. A hybrid gpu cluster and volunteer computing platform for scalable deep learning. *The Journal of Supercomputing*, 04 2018.

[36] Max Ryabinin and Anton Gusev. Towards crowdsourced training of large neural networks using decentralized mixture-of-experts. In *Advances in Neural Information Processing Systems*, 2020.

[37] Aaron Harlap, Alexey Tumanov, Andrew Chung, Gregory R. Ganger, and Phillip B. Gibbons. Proteus: Agile ml elasticity through tiered reliability in dynamic resource markets. In *Proceedings of the Twelfth European Conference on Computer Systems*, EuroSys '17, page 589–604, New York, NY, USA, 2017. Association for Computing Machinery.

[38] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE transactions on information theory*, 52(6):2508–2530, 2006.

[39] John Nikolas Tsitsiklis. Problems in decentralized decision making and computation. Technical report, Massachusetts Inst of Tech Cambridge Lab for Information and Decision Systems, 1984.

[40] Kevin Scaman, Francis Bach, Sébastien Bubeck, Yin Tat Lee, and Laurent Massoulié. Optimal algorithms for smooth and strongly convex distributed optimization in networks. In *International Conference on Machine Learning*, pages 3027–3036, 2017.

[41] Kevin Scaman, Francis Bach, Sébastien Bubeck, Laurent Massoulié, and Yin Tat Lee. Optimal algorithms for non-smooth distributed optimization in networks. In *Advances in Neural Information Processing Systems*, pages 2740–2749, 2018.

[42] Kevin Scaman, Francis Bach, Sébastien Bubeck, Yin Lee, and Laurent Massoulié. Optimal convergence rates for convex distributed optimization in networks. *Journal of Machine Learning Research*, 20:1–31, 2019.

[43] Mahmoud Assran, Nicolas Loizou, Nicolas Ballas, and Mike Rabbat. Stochastic gradient push for distributed deep learning. In *International Conference on Machine Learning*, pages 344–353. PMLR, 2019.

[44] Lin Xiao and Stephen Boyd. Fast linear iterations for distributed averaging. *Systems & Control Letters*, 53(1):65–78, 2004.

[45] Russell Merris. Laplacian matrices of graphs: a survey. *Linear algebra and its applications*, 197:143–176, 1994.

[46] César A Uribe, Soomin Lee, Alexander Gasnikov, and Angelia Nedić. A dual approach for optimal algorithms in distributed optimization over networks. *Optimization Methods and Software*, pages 1–40, 2020.

[47] Angelia Nedić and Alex Olshevsky. Distributed optimization over time-varying directed graphs. *IEEE Transactions on Automatic Control*, 60(3):601–615, 2014.

[48] Angelia Nedić and Alex Olshevsky. Stochastic gradient-push for strongly convex functions on time-varying directed graphs. *IEEE Transactions on Automatic Control*, 61(12):3936–3947, 2016.

[49] Angelia Nedić, Alex Olshevsky, and Michael G Rabbat. Network topology and communication-computation tradeoffs in decentralized optimization. *Proceedings of the IEEE*, 106(5):953–976, 2018.

[50] Alexander Rogozin and Alexander Gasnikov. Projected gradient method for decentralized optimization over time-varying networks. *arXiv preprint arXiv:1911.08527*, 2019.

[51] S Sundhar Ram, A Nedić, and Venugopal V Veeravalli. Asynchronous gossip algorithms for stochastic optimization. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 3581–3586. IEEE, 2009.

[52] Feng Yan, Shreyas Sundaram, SVN Vishwanathan, and Yuan Qi. Distributed autonomous online learning: Regrets and intrinsic privacy-preserving properties. *IEEE Transactions on Knowledge and Data Engineering*, 25(11):2483–2493, 2012.

[53] Kun Yuan, Qing Ling, and Wotao Yin. On the convergence of decentralized gradient descent. *SIAM Journal on Optimization*, 26(3):1835–1854, 2016.

[54] Paul Sack and William Gropp. Collective algorithms for multiported torus networks. *ACM Trans. Parallel Comput.*, 1(2), February 2015.

[55] Petar Maymounkov and David Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *International Workshop on Peer-to-Peer Systems*, pages 53–65. Springer, 2002.

[56] Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4):1574–1609, 2009.

[57] Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

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

[59] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[60] Eduard Gorbunov, Filip Hanzely, and Peter Richtarik. Local sgd: Unified theory and new efficient methods. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 3556–3564. PMLR, 13–15 Apr 2021.

[61] Ahmed Khaled, Konstantin Mishchenko, and Peter Richtárik. Tighter theory for local sgd on identical and heterogeneous data. In *International Conference on Artificial Intelligence and Statistics*, pages 4519–4529. PMLR, 2020.

[62] Blake Woodworth, Kumar Kshitij Patel, Sebastian Stich, Zhen Dai, Brian Bullins, Brendan Mcmahan, Ohad Shamir, and Nathan Srebro. Is local sgd better than minibatch sgd? In *International Conference on Machine Learning*, pages 10334–10343. PMLR, 2020.

[63] Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian Stich. A unified theory of decentralized sgd with changing topology and local updates. In *International Conference on Machine Learning*, pages 5381–5393. PMLR, 2020.

[64] Xiang Li, Wenhao Yang, Shusen Wang, and Zhihua Zhang. Communication efficient decentralized training with multiple local updates. *arXiv preprint arXiv:1910.09126*, 5, 2019.

[65] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015.

[66] Xiangru Lian, Wei Zhang, Ce Zhang, and Ji Liu. Asynchronous decentralized parallel stochastic gradient descent. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3043–3052. PMLR, 10–15 Jul 2018.

[67] Andrei M Sukhov, MA Astrakhantseva, AK Pervitsky, SS Boldyrev, and AA Bukatov. Generating a function for network delay. *Journal of High Speed Networks*, 22(4):321–333, 2016.

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

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

[70] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

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

[72] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015.

[73] Jiahuang Lin, Xin Li, and Gennady Pekhimenko. Multi-node bert-pretraining: Cost-efficient approach, 2020.

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

[75] NVIDIA. Nvidia data center deep learning product performance. "`https://developer.nvidia.com/deep-learning-performance-training-inference`", accessed at 2021.02.03.

[76] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and Hugh Brendan McMahan. Adaptive federated optimization. In *International Conference on Learning Representations*, 2021.

[77] Xiangyi Chen, Xiaoyun Li, and Ping Li. Toward communication efficient adaptive gradient method. In *Proceedings of the 2020 ACM-IMS on Foundations of Data Science Conference*, FODS '20, page 119–128, New York, NY, USA, 2020. Association for Computing Machinery.

[78] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*, 2016.

[79] David Aldous and James Allen Fill. Reversible markov chains and random walks on graphs, 2002. unfinished monograph, recompiled 2014, 2002.

[80] Jinming Xu, Ye Tian, Ying Sun, and Gesualdo Scutari. Distributed algorithms for composite optimization: Unified and tight convergence analysis. *arXiv preprint arXiv:2002.11534*, 2020.

[81] Alireza Fallah, Mert Gurbuzbalaban, Asu Ozdaglar, Umut Simsekli, and Lingjiong Zhu. Robust distributed accelerated stochastic gradient methods for multi-agent networks. *arXiv preprint arXiv:1910.08701*, 2019.

[82] Dmitry Kovalev, Adil Salim, and Peter Richtárik. Optimal and practical algorithms for smooth and strongly convex decentralized optimization. *Advances in Neural Information Processing Systems*, 33, 2020.

[83] Yossi Arjevani and Ohad Shamir. Communication complexity of distributed convex learning and optimization. *Advances in neural information processing systems*, 28:1756–1764, 2015.

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

[85] Dan Alistarh, Demjan Grubic, Jerry Z Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: communication-efficient sgd via gradient quantization and encoding. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1707–1718, 2017.

[86] Ananda Theertha Suresh, X Yu Felix, Sanjiv Kumar, and H Brendan McMahan. Distributed mean estimation with limited communication. In *International Conference on Machine Learning*, pages 3329–3337. PMLR, 2017.

[87] Ali Ramezani-Kebrya, Fartash Faghri, Ilya Markov, Vitalii Aksenov, Dan Alistarh, and Daniel M Roy. Nuqsgd: Provably communication-efficient data-parallel sgd via nonuniform quantization. *Journal of Machine Learning Research*, 22(114):1–43, 2021.

[88] Fartash Faghri, Iman Tabrizian, Ilia Markov, Dan Alistarh, Daniel M Roy, and Ali Ramezani-Kebrya. Adaptive gradient quantization for data-parallel sgd. *Advances in Neural Information Processing Systems*, 33:3174–3185, 2020.

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

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

[91] Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Terngrad: ternary gradients to reduce communication in distributed deep learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1508–1518, 2017.

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

[93] Samuel Horváth, Dmitry Kovalev, Konstantin Mishchenko, Sebastian Stich, and Peter Richtárik. Stochastic distributed learning with gradient quantization and variance reduction. *arXiv preprint arXiv:1904.05115*, 2019.

[94] Zhize Li, Dmitry Kovalev, Xun Qian, and Peter Richtarik. Acceleration for compressed gradient descent in distributed and federated optimization. In *International Conference on Machine Learning*, pages 5895–5904. PMLR, 2020.

[95] Eduard Gorbunov, Dmitry Kovalev, Dmitry Makarenko, and Peter Richtarik. Linearly converging error compensated sgd. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 20889–20900. Curran Associates, Inc., 2020.

[96] Constantin Philippenko and Aymeric Dieuleveut. Artemis: tight convergence guarantees for bidirectional compression in federated learning. *arXiv preprint arXiv:2006.14591*, 2020.

[97] Zhize Li and Peter Richtárik. A unified analysis of stochastic gradient methods for nonconvex federated optimization. *arXiv preprint arXiv:2006.07013*, 2020.

[98] Farzin Haddadpour, Mohammad Mahdi Kamani, Aryan Mokhtari, and Mehrdad Mahdavi. Federated learning with compression: Unified analysis and sharp guarantees. *arXiv preprint arXiv:2007.01154*, 2020.

[99] Rudrajit Das, Abolfazl Hashemi, Sujay Sanghavi, and Inderjit S Dhillon. Improved convergence rates for non-convex federated learning with compression. *arXiv preprint arXiv:2012.04061*, 2020.

[100] Eduard Gorbunov, Konstantin P. Burlachenko, Zhize Li, and Peter Richtarik. Marina: Faster non-convex distributed learning with compression. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 3788–3798. PMLR, 18–24 Jul 2021.

[101] Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified sgd with memory. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 4452–4463, 2018.

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

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

[104] Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, and Ramtin Pedarsani. An exact quantized decentralized gradient descent algorithm. *IEEE Transactions on Signal Processing*, 67(19):4934–4947, 2019.

[105] Dmitry Kovalev, Anastasia Koloskova, Martin Jaggi, Peter Richtarik, and Sebastian Stich. A linearly convergent algorithm for decentralized optimization: Sending less bits for free! In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 4087–4095. PMLR, 13–15 Apr 2021.

[106] Anastasia Koloskova, Tao Lin, Sebastian U Stich, and Martin Jaggi. Decentralized deep learning with arbitrary communication compression. In *International Conference on Learning Representations*, 2020.

[107] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

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

[109] Sebastian Urban Stich. Local SGD converges fast and communicates little. *International Conference on Learning Representations (ICLR)*, page arXiv:1805.09767, 2019.

[110] Tao Lin, Sebastian Urban Stich, Kumar Kshitij Patel, and Martin Jaggi. Don't use large mini-batches, use local SGD. *ICLR*, page arXiv:1808.07217, 2020.

[111] Blake Woodworth, Kumar Kshitij Patel, and Nathan Srebro. Minibatch vs local sgd for heterogeneous distributed learning. *arXiv preprint arXiv:2006.04735*, 2020.

[112] Honglin Yuan and Tengyu Ma. Federated accelerated stochastic gradient descent. *Advances in Neural Information Processing Systems*, 33, 2020.

[113] Debraj Basu, Deepesh Data, Can Karakus, and Suhas Diggavi. Qsparse-local-SGD: Distributed SGD with quantization, sparsification and local computations. In *Advances in Neural Information Processing Systems*, pages 14668–14679, 2019.

[114] Honglin Yuan, Manzil Zaheer, and Sashank Reddi. Federated composite optimization. *arXiv preprint arXiv:2011.08474*, 2020.

[115] Mahmoud Assran, Arda Aytekin, Hamid Reza Feyzmahdavian, Mikael Johansson, and Michael G Rabbat. Advances in asynchronous parallel and distributed optimization. *Proceedings of the IEEE*, 108(11):2013–2031, 2020.

[116] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in neural information processing systems*, pages 693–701, 2011.

[117] Shen-Yi Zhao and Wu-Jun Li. Fast asynchronous parallel stochastic gradient descent: A lock-free approach with convergence guarantee. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[118] Rémi Leblond, Fabian Pedregosa, and Simon Lacoste-Julien. Asaga: asynchronous parallel saga. In *Artificial Intelligence and Statistics*, pages 46–54. PMLR, 2017.

[119] Zhimin Peng, Yangyang Xu, Ming Yan, and Wotao Yin. Arock: an algorithmic framework for asynchronous parallel coordinate updates. *SIAM Journal on Scientific Computing*, 38(5):A2851–A2879, 2016.

[120] Konstantin Mishchenko, Franck Iutzeler, Jérôme Malick, and Massih-Reza Amini. A delay-tolerant proximal-gradient algorithm for distributed learning. In *International Conference on Machine Learning*, pages 3587–3595. PMLR, 2018.

[121] Alekh Agarwal and John C Duchi. Distributed delayed stochastic optimization. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, pages 873–881, 2011.

[122] Hamid Reza Feyzmahdavian, Arda Aytekin, and Mikael Johansson. An asynchronous mini-batch algorithm for regularized stochastic optimization. *IEEE Transactions on Automatic Control*, 61(12):3740–3754, 2016.

[123] Yossi Arjevani, Ohad Shamir, and Nathan Srebro. A tight convergence analysis for stochastic gradient descent with delayed updates. In *Algorithmic Learning Theory*, pages 111–132. PMLR, 2020.

[124] Hari Balakrishnan, M Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Looking up data in p2p systems. *Communications of the ACM*, 46(2):43–48, 2003.

[125] Seymour Kaplan. Application of programs with maximin objective functions to problems of optimal resource allocation. *Operations Research*, 22(4):802–807, 1974.

[126] Erling D. Andersen and Knud D. Andersen. The mosek interior point optimizer for linear programming: An implementation of the homogeneous algorithm. In *Applied Optimization*, pages 197–232. Springer US, 2000.

[127] Anand Jayarajan, Jinliang Wei, Garth Gibson, Alexandra Fedorova, and Gennady Pekhimenko. Priority-based parameter propagation for distributed dnn training. In A. Talwalkar, V. Smith, and M. Zaharia, editors, *Proceedings of Machine Learning and Systems*, volume 1, pages 132–145, 2019.

Supplementary Material

A GPU instance costs

This section provides a brief cost analysis of typical deep learning compute resources both in the cloud and on-premises. For brevity, we limit this analysis to the popular GPUs available at the time of submission. Note that the exact costs will depend on a variety of factors such as the cloud provider, the region, electricity costs, and market fluctuations. Therefore, we warn the reader to consider this analysis only as a rough estimate.

Specifically, we estimate the compute costs for the occasional usage scenario: running a single set of experiments over several weeks or conducting infrequent experiments. This scenario covers most research scientists and small organizations. The most straightforward way to provision a GPU server in such a scenario is to rent it from a cloud provider (e.g., GCP or AWS) or a public marketplace (e.g., Vast.ai or Golem).

While the exact server specifications vary from one provider to another, there are two broad categories of GPU machines: regular and preemptible. Regular instance types typically offer 1–8 GPUs per node with tight uptime guarantees (typically 99.99%) and a high-bandwidth network (tens of Gb/s). In turn, preemptible instances provide the same resource type at a significant discount with the condition that the machine can be terminated at any time after short notice.

To account for individual variations, we report the average rent price over three popular cloud providers. We consider three popular instance types: two high-end instances with 8 Tesla V100 or A100 GPUs and a low-end instance with a single Tesla T4 GPU. We also describe several low-end servers and workstations available on a public marketplace. Unlike cloud VMs, these instances are hosted on non-curated hardware with less uptime guarantees (typically 95% – 99.9%), slower network and significant variation in performance. However, marketplace instances are the cheapest in terms of cost per TFLOPS. To quantify this, we report the average over three most affordable instances that fit the chosen minimum requirements.

As a point of comparison, we also measure each system's training performance for BERT-Large [68] fine-tuning on SQuAD v1.1 [78] in PyTorch with mixed precision. We follow the official benchmarking protocol by [75] and reuse the official performance results for V100, A100, and T4 instances. The only exception is GTX 1080Ti, where we use full 32-bit precision because that device does not support efficient half-precision operations.

Table 1: Cloud and marketplace GPU instance pricing for short-term usage.

Minimum system specifications				Average cost, $/hour		BERT-Large
GPU	CPU cores	CPU type	RAM, GB	Regular	Preemptible	training samples/s
Cloud instances						
8× V100	64	Intel Xeon Broadwell	480	23.47	7.13	354
8× A100	96	AMD Epyc ROME	960	30.65	10.18	755
1× T4	4	Intel Xeon Cascade Lake	16	0.46	0.18	18
Marketplace instances						
6× 3090	32	AMD Epyc Rome	480	5.04	4.17	154
4× 2080Ti	16	Intel Xeon Haswell	240	0.96	0.84	83.4
1× RTX 1080Ti	8	Intel Xeon Haswell	16	0.22	0.16	12

Table 1 shows two main tendencies. First, preemptible *cloud* instances are, on average, three times cheaper than their non-preemptible counterparts[8]. Second, the high-end HPC-grade servers that offer the highest raw performance are less cost-effective than lower-tier servers and marketplace instances. In theory, one could match the raw floating-point performance of a 8×V100 instance at a fraction of its cost using multiple lower-tier workstations, such as 4× RTX 2080Ti, with a smaller total cost.

[8] The cost can be up to 11× cheaper for some instance types, e.g. Azure V100 instances in the central US region at the time of writing.

However, in practice, running distributed training with these workstations is challenging due to their unreliability and slow network connection.

Note that this analysis does not represent the cloud costs for sustained GPU usage. If an organization plans to constantly use GPU resources over a period of multiple years, they can reduce the costs by deploying their own compute infrastructure or relying on the sustained usage discounts reaching up to 60–70%. Thus, the long-term compute costs are much harder to analyze and depend on a number of additional factors, such as local electricity prices for on-premise infrastructure. However, this scenario offers similar trade-offs: HPC-grade infrastructure offers greater interconnectivity, but requires expensive network interface cards, high-end switches and a more complex setup process.

B Additional Related Work

In this section, we review some of the papers relevant to our work, but omitted from the main part due to space constraints.

B.1 Decentralized training

In this subsection, we give additional details about the dependence of gossip-based optimization methods on the spectral properties on the communication graph through the spectral properties of the mixing matrix [44, 42] or the Laplacian matrix [45, 46] of the network. That is, gossip finds approximate average on nodes with accuracy ε after $\mathcal{O}\left((1-\lambda_2(\mathbf{M}))^{-1}\log(\varepsilon^{-1})\right)$ iterations, where $\mathbf{M}$ is the mixing matrix and $\lambda_2(\mathbf{M})$ is the second largest eigenvalue of $\mathbf{M}$ when sorted by absolute value. The quantity $\eta = 1-\lambda_2(\mathbf{M})$ is called the spectral gap of the mixing matrix $\mathbf{M}$, and η^{-1} is typically a polynomial of the total number of nodes N when the maximal degree of the node is $\mathcal{O}(1)$. For example, for uniformly averaging $\mathbf{M}$ one can show that $\eta^{-1} = \mathcal{O}(N^2)$ for the ring topology (node degree 2), $\eta^{-1} = \mathcal{O}(N)$ for the two-dimensional torus topology (node degree 2), and $\eta^{-1} = \mathcal{O}(1)$ for the fully connected graph (node degree $N-1$); one can find more examples in [79]. Similarly, the communication complexity of decentralized optimization methods often has multiplicative dependence on either $\mathcal{O}(\eta^{-1})$ (see [80] and references therein) or $\mathcal{O}(\eta^{-1/2})$ [42, 46, 81, 82], which is not improvable for gossip-based methods [83, 40].

Contrary to this, Moshpit All-Reduce does not depend on a fixed communication graph and the properties of its mixing matrix. However, it depends on the number of averaging groups and the total number of peers (see Theorem 3.2), which can be viewed as properties of a time-varying random communication graph. Fortunately, this dependence is often much better than in gossip: as we mentioned in the main part of the paper, even if workers are randomly split into pairs at each iteration, the simplified version of Moshpit All-Reduce makes the average distortion (the left-hand side of Equation 5) at least 2 times smaller after each round on average.

B.2 Compressed communication

Another popular approach to address the communication bottleneck is communication compression [84, 85, 86, 87, 88]: before sending any information (e.g., iterates, gradients, Hessians or more sophisticated data) over the network, peers compress this information by applying a possibly random transformation. As the result, peers send fewer bits for each communication round, but the total number of communication rounds needed to achieve the predefined accuracy of the solution increases. However, compression can be useful in situations when the reduction in communication costs of one round is more important than the increase in the number of these rounds [89].

There are two distinct groups of works on distributed training with compressed communication: ones that focus on unbiased compression operators (e.g., Rand-K, ℓ_p-quantization) and ones studying algorithms with biased compressors (e.g., Top-K); see a detailed summary of popular compression operators in [90]). Quantized SGD (QSGD) [85] and TernGrad [91] were among the first compression methods with convergence guarantees. Next, the convergence analysis of these methods was generalized and tightened in the (strongly) convex case in [92]. Moreover, the authors of [92] proposed a modification of QSGD called DIANA: this algorithm is based on the quantization of gradients' differences, which helps it achieve linear convergence in the strongly convex case when peers compute full gradients. Next, DIANA was generalized to arbitrary unbiased compression in [93], where authors also developed and analyzed the variance-reduced version of DIANA. After that, several further modifications, such as Accelerated DIANA [94] and DIANA with bidirectional compression [95, 96], were proposed. Finally, we refer the reader to [97, 98, 99, 100] for state-of-the-art results for distributed methods with unbiased compression in the non-convex case.

However, naïve application of biased compression operators can lead to significantly worse performance in practice. For instance, as it was shown recently in [90], parallel SGD with Top-1 compression can diverge exponentially fast. Therefore, biased compressors are used jointly with so-called error-compensation [84]. The first analysis of Error-Compensated SGD (EC-SGD) was proposed in [101, 102] which then was generalized and tightened in [90]. Next, several further improvements, such as an accelerated version of EC-SGD [103] and linearly converging EC-SGD [95], were recently proposed. However, current theory does not show any superiority of distributed methods with biased compressors to the ones with unbiased compression operators. In addition, one can combine decentralized communication with compression. Such combinations with unbiased compression operators were studied in [104, 105] and with biased operators in [24, 106]. In this paper, we do not study the interaction of different compression methods and Moshpit Averaging, leaving this promising direction to future work.

B.3 Multiple local steps

Alternatively, to reduce the impact of the communication bottleneck, it is possible to perform several local optimization steps on each peer between the communication rounds. This approach is based on the idea that the increased computational load of peers will decrease the number of communication rounds required to obtain the optimal parameters; it is frequently used in federated learning [107, 108]. In particular, one of the most popular methods with multiple local steps is called Local-SGD or Federated Averaging [107, 109]. The first results on its convergence were given in [109, 110], and later they were tightened and generalized both for homogeneous [61, 62] and heterogeneous cases [61, 111]. Recently, further modifications of Local-SGD were proposed and analyzed: these modifications include acceleration [112], variance reduction [60], communication compression [113, 98, 99], decentralization [64, 63], adaptive and proximal methods [76, 114], and resistance to client drift [59]. Moshpit SGD can perform multiple local gradient steps before synchronization by design, as shown in Algorithm 2.

B.4 Asynchronous methods

In the previous subsections, we mostly discussed synchronous distributed methods, since they are more widespread and better studied than asynchronous ones. Mainly, this is because asynchronous methods are more difficult to implement, debug and analyze under general assumptions. However, such methods can be more efficient in terms of using computational resources, which leads to faster wall-clock convergence [115]. In recent years, several asynchronous stochastic methods [116, 117, 118], methods with no shared memory [119, 120], and methods with delayed updates [121, 122, 123, 95] were proposed and analyzed: one can find more details in a recent survey [115]. Moshpit SGD belongs to this family of asynchronous approaches as well, because the averaging steps happen in smaller groups and can be interleaved with local parameter updates.

B.5 Distributed Hash Tables

In this work, we set out to improve distributed averaging with a dynamic matchmaking protocol. Without a central server, this protocol relies on decentralized data structures to organize peers. The main data structure we use is the Distributed Hash Table, or DHT. On a high level, DHT is a distributed fault-tolerant "dictionary" that can be accessed by every participant. Each key-value pair is stored on a subset of peers determined by the hash function of the key.

Each participant has a unique identifier (ID) sampled uniformly from the hash function output range. When storing a $(key,\ value)$ pair, one must find k peers whose IDs are nearest to $\mathrm{hash}(key)$ according to a chosen metric. After that, the participant requests each of those peers to store $(key,\ value)$. When retrieving a value for a key, one should compute $\mathrm{hash}(key)$, search for peers with IDs nearest to that hash value and request the value from those peers.

Specific DHT versions, such as Chord [124] or Kademlia [55], employ different hash types and algorithms for finding nearest peers. For instance, Kademlia DHT sorts peers based on the XOR distance function: $d(x, y) = \mathrm{int}(x \oplus y)$.

In DHT, each participant is directly aware of only a small subset of peers. When storing or retrieving a key, the participant requests additional peers from its neighbors in a semi-greedy search, minimizing the XOR distance until it finds k nearest peers. In Kademlia, nodes form a special navigable graph structure that lets them find nearest peers in at most $\mathcal{O}(k + \log N)$ requests to other peers, where N is the total number of participants. Due to their scalability and fault-tolerance, DHTs found numerous applications including BitTorrent, Ethereum, I2P and decentralized deep learning [36].

C Proofs of Mixing Properties of Moshpit All-Reduce

Notation. Throughout the following sections, we use the standard notation from the literature on stochastic optimization. That is, for any n-dimensional vectors $x = (x_1, \ldots, x_n)^\top, y = (y_1, \ldots, y_n)^\top \in \mathbb{R}^n$ we use $\langle x, y\rangle$ to denote the standard inner product: $\langle x, y\rangle = x_1y_1 + \ldots + x_ny_n$. Next, we use $\|x\|$ to denote the ℓ_2=norm of x ($\|x\| = \sqrt{\langle x, x\rangle}$), $\mathbb{E}[\xi]$ to denote an expectation of a random variable ξ, $\mathbb{E}[\xi \mid \eta]$ is used for the conditional expectation of ξ given η, and $\mathbb{P}\{E\}$ denotes the probability of an event E.

C.1 Computing exact average in a full grid

As discussed in Section 3.1, Moshpit All-Reduce obtains the exact average of parameter vectors from N peers arranged in a grid with d coordinates and M positions per coordinate when $N \equiv M^d$. That is, when the grid is full and each step averages M parameter values along a single grid coordinate without repetitions, the algorithm needs only d steps to compute the actual average across all nodes. In this section, we give a proof of this fact.

First, let us formally define the setting and the averaging steps of Moshpit All-Reduce in this specific case. Let $\theta_{i_1 i_2 \ldots i_d}$ be the parameter vector of the worker with coordinates $i_1, i_2, \ldots, i_d$; each coordinate i_k takes values from 1 to M, because the hypercube of peers is completely full (thus, due to the pigeonhole principle, there are no unoccupied coordinates). Next, arrange the coordinates of these vector according to the order of averaging iterations: namely, at iteration 1

$$\overline{\theta}^1_{i_1 i_2 \ldots i_d} = \frac{1}{M}\sum_{j_1=1}^{M} \theta_{j_1 i_2 \ldots i_d}, \quad i_1 \in \{1, \ldots, M\}, \tag{13}$$

which means that for the first iteration, we take the average across the first axis $\overline{\theta}^1$ and replicate it across all M resulting vectors regardless of their index i_1. The next averaging steps can be expressed similarly with a simple recurrence relation:

$$\overline{\theta}^t_{i_1 i_2 \ldots i_d} = \frac{1}{M}\sum_{j_t=1}^{M} \overline{\theta}^{t-1}_{i_1 \ldots i_{t-1} j_t i_{t+1} \ldots i_d}. \tag{14}$$

Given this formal definition, we can now state and prove the exact averaging result:

Theorem C.1 (Exact average in a full d-dimensional hypercube after d steps). *Assume that M^d peers are arranged in a d-dimensional hypercube with M positions in each dimension. Also, assume that each peer fully participates in every averaging step and M-sized groups for each averaging iteration are determined based on the hypercube coordinates. Then, if Moshpit All-Reduce is ran in the above setup for d iterations without repeating groups (i.e. averaging across each dimension exactly once), its result for each participant is the average value of θ across all M^d peers.*

Proof. We can directly obtain the expression for the average by expanding the recurrence and rearranging the sums:

$$\begin{aligned}
\overline{\theta}^d_{i_1 i_2 \dots i_d} &= \frac{1}{M}\sum_{j_d=1}^{M} \overline{\theta}^{d-1}_{i_1 \dots i_{d-1} j_d} = \frac{1}{M}\sum_{j_d=1}^{M}\left(\frac{1}{M}\sum_{j_{d-1}=1}^{M} \overline{\theta}_{i_1 i_2 \dots j_{d-1} j_d}\right) = \dots \\
&= \frac{1}{M}\Bigg(\underbrace{\sum_{j_d=1}^{M}\Bigg(\frac{1}{M}\sum_{j_{d-1}=1}^{M} \dots \sum_{j_2=1}^{M}\Bigg(\frac{1}{M}\sum_{j_1=1}^{M}}_{d \text{ summations}} \theta_{j_1 \dots j_d}\Bigg)\Bigg)\Bigg) \\
&= \frac{1}{M^d}\sum_{j_d=1}^{M}\sum_{j_{d-1}=1}^{M} \dots \sum_{j_2=1}^{M}\sum_{j_1=1}^{M} \theta_{j_1 \dots j_d} = \frac{1}{M^d}\sum_{j_1, \dots, j_d=1}^{M} \theta_{j_1 \dots j_d}.
\end{aligned}$$

But this is exactly the global average of all θ, since there are M^d participants and each vector is represented in the sum because of summation over all possible indices. □

Notice that for a given grid of peers, if some of its indices do not have corresponding parameter vectors, Equation 14 may result in different average vectors on different workers due to different numbers of peers along a coordinate for different indices. For example, running two iterations of Moshpit Averaging with $d = 2,\ M = 2$ and three parameter vectors $\theta_{11},\ \theta_{21},\ \theta_{22}$ results in $\frac{\theta_{11}+\theta_{21}}{2}$ on the first worker and $\frac{\theta_{11}+\theta_{21}}{4} + \theta_{22}$ on other workers, with neither equal to the global average. However, the variance of the averaged vectors does decrease, which is formally proven in Section C.3.

C.2 Proof of Theorem 3.1

Below we provide the complete proof of Theorem 3.1. For the readers' convenience, we restate the theorem.

Theorem C.2 (Theorem 3.1)**.** *If all workers have non-zero probability of successfully running a communication round in Moshpit Averaging and the order of $\texttt{peers}_t$ is random, then all local vectors θ_i^t converge to the global average with probability* 1*:*

$$\forall i = 1, \dots, N \quad \left\|\theta_i^t - \frac{1}{N}\sum_{i=1}^{N} \theta_i^0\right\|^2 \xrightarrow[t\to\infty]{} 0. \tag{15}$$

Proof of Theorem 3.1. First of all, we notice that (15) is equivalent to

$$\forall i = 1, \dots, N,\ \forall j = 1, \dots, n \quad \left(\theta_i^t(j) - \frac{1}{N}\sum_{i=1}^{N} \theta_i^0(j)\right)^2 \xrightarrow[t\to\infty]{} 0, \tag{16}$$

where $\theta_i^t(j)$ denotes j-th component of θ_i^t. Consider an arbitrary component $j \in \{1, \dots, n\}$ and the sequence of intervals $\{I_{j,t}\}_{t\geq 0}$ where $I_{j,t} = \operatorname{conv}\{\theta_1^t(j), \theta_2^t(j), \dots, \theta_N^t(j)\}$. Then, $\{I_{j,t}\}_{t\geq 0}$ is a sequence of nested intervals ($I_{j,t+1} \subseteq I_{j,t} \forall t \geq 0$), since averaging in groups does not expand the convex hull of $\{\theta_1^t, \theta_2^t, \dots, \theta_N^t\}$. For convenience, we specify the bounds of the intervals: $I_{j,t} = [a_{j,t}, b_{j,t}]$. Using the Cantor's intersection theorem, we conclude that

$$\bigcap_{t=0}^{\infty} I_{j,t} = I_j = [a_j, b_j],$$

where $\overline{\theta}(j) = \frac{1}{N}\sum_{i=1}^{n} \theta_i^0(j) \in [a_j, b_j]$. If $[a_j, b_j] = \{\overline{\theta}(j)\}$ with probability 1, then (16) holds with probability 1 as well. Suppose the opposite: there exist such $j \in \{1, \dots, n\}$, $[a, b]$ and $\delta, \Delta > 0$ that $\overline{\theta}(j) \in [a, b]$, $b - a = \Delta$ and

$$\mathbb{P}\Bigg\{\underbrace{[a, b] \subseteq \bigcap_{t=0}^{\infty} I_{j,t}}_{E}\Bigg\} = \delta > 0 \quad \text{and} \quad \forall \varepsilon > 0\ \mathbb{P}\Bigg\{\underbrace{[a-\varepsilon, b+\varepsilon] \subseteq \bigcap_{t=0}^{\infty} I_{j,t}}_{E_\varepsilon}\Bigg\} < \delta.$$

This implies that for all $\varepsilon > 0$ there exists such $T_\varepsilon > 0$ that

$$\mathbb{P}\Big\{ \underbrace{\forall t \geq T_\varepsilon \;\; a_{j,t} \in [a-\varepsilon, a], b_{j,t} \in [b, b+\varepsilon]}_{E'_\varepsilon} \Big\} = \delta_\varepsilon > 0.$$

Consider $\varepsilon = \frac{\Delta}{(2N+100)^{2N}}$ and assume that the event E'_ε holds. Next, we introduce new notation: $J^t_{\text{left}} = \{i \in \{1,\ldots,n\} \mid \theta^t_i(j) \in [a-\varepsilon, a]\}$ and $J^t_{\text{right}} = \{i \in \{1,\ldots,n\} \mid \theta^t_i(j) \in [b, b+\varepsilon]\}$. Since E'_ε holds the sets J^t_{left} and J^t_{right} are non-empty for all $t \geq T_\varepsilon$ with probability $\delta_\varepsilon > 0$:

$$\mathbb{P}\left\{\forall t \geq T_\varepsilon \;\; J^t_{\text{left}} \neq \varnothing \text{ and } J^t_{\text{right}} \neq \varnothing\right\} = \delta_\varepsilon > 0. \tag{17}$$

We notice that every pair of workers i_1, i_2 has a non-zero probability of taking part in the averaging inside the common group at each iteration since all workers have a non-zero probability of successfully running a communication round and the order of $\texttt{peers}_t$ is random. This implies that every pair of workers i_1, i_2 with probability 1 take part in the averaging inside the common group infinitely many times when t goes to the infinity.

Next, we choose some $t_0 \geq T_\varepsilon$. Let $J^{t_0}_{\text{left}} = \{i_{l,1}, \ldots, i_{l,q_l}\}$ and $J^{t_0}_{\text{right}} = \{i_{r,1}, \ldots, i_{r,q_r}\}$. Consider the event $E'_{\varepsilon,0} \subseteq E'_\varepsilon$ such that in $E'_{\varepsilon,0}$ peer $i_{l,1}$ computes an average in the group containing any peer from $J^{t_0}_{\text{right}}$ at some iteration $t_1 > t_0$. Our observations above imply that $\mathbb{P}\{E'_{\varepsilon,0}\} = \mathbb{P}\{E'_\varepsilon\} = \delta_\varepsilon > 0$. Then, $\theta^{t_1}_{i_{l,1}}(j) \geq \frac{N-1}{N}(a-\varepsilon) + \frac{1}{N}b = a - \varepsilon + \frac{1}{N}(\Delta+\varepsilon) = a - \frac{\Delta}{(2N+100)^{2N}} + \frac{1}{N}\left(\Delta + \frac{\Delta}{(2N+100)^{2N}}\right) > a + \frac{\Delta}{2N}$, i.e., $\theta^{t_1}_{i_{l,1}}(j) \in (a, b]$ meaning that $i_{l,1} \notin J^{t_1}_{\text{left}}$. The last part of the proof shows that for any $t \geq t_1$, the peer $i_{l,1}$ will never be the part of J^t_{left} and after a finite number of iterations $J^t_{\text{left}} = \varnothing$ with probability $\delta_\varepsilon > 0$ when $E'_{\varepsilon,0}$ holds, implying the contradiction with (17).

To show that, we consider the following set of peers: $\widehat{J}^{t_1}_{\text{left}} = \{i \in \{1,\ldots,n\} \mid \exists t \geq t_1 : \; \theta^t_i(j) \in [a-\varepsilon, a + \frac{\Delta}{2N})\}$. Next, we consider the event $E'_{\varepsilon,1} \subseteq E'_{\varepsilon,0}$ such that in $E'_{\varepsilon,1}$ peer $i_{l,1}$ computes an average in the group containing some peer $i_{l,avg,1}$ from $\widehat{J}^{t_1}_{\text{left}}$ at some iteration $t_2 > t_1$ (and t_2 is the first such moment after t_1). Again, our observations imply $\mathbb{P}\{E'_{\varepsilon,1}\} = \mathbb{P}\{E'_{\varepsilon,0}\} = \delta_\varepsilon > 0$. Then, $\theta^{t_2}_{i_{l,1}}(j) = \theta^{t_2}_{i_{l,avg,1}}(j) > \frac{N-1}{N}(a-\varepsilon) + \frac{1}{N}\left(a + \frac{\Delta}{2N}\right) = a + \frac{\Delta}{2N^2} - \frac{(N-1)\Delta}{N(2N+100)^{2N}} > a + \frac{\Delta}{4N^2}$. After that, we consider the event $E'_{\varepsilon,2} \subseteq E'_{\varepsilon,1}$ such that in $E'_{\varepsilon,2}$ peer $i_{l,1}$ or $i_{l,avg,1}$ computes an average in the group containing a peer $i_{l,avg,2} \neq i_{l,avg,1}$ from $\widehat{J}^{t_1}_{\text{left}}$ at an iteration $t_3 > t_2$ (and t_3 is the first such moment after t_2). Then, $\theta^{t_3}_{i_{l,1}}(j), \theta^{t_3}_{i_{l,avg,1}}(j)$ and $\theta^{t_3}_{i_{l,avg,2}}(j)$ are greater than $\frac{N-1}{N}(a-\varepsilon) + \frac{1}{N}\left(a + \frac{\Delta}{4N^2}\right) = a + \frac{\Delta}{4N^3} - \frac{(N-1)\Delta}{N(2N+100)^{2N}} > a + \frac{\Delta}{8N^3}$.

Therefore, after at least $N-1$ of such averaging iterations, with probability δ_ε all $\theta^t_i(j)$ will be greater than $a + \frac{\Delta}{(2N)^N} > a$ while E'_ε holds. This contradicts (17). Therefore,

$$\bigcap_{t=0}^{\infty} I_{j,t} = \{\overline{\theta}(j)\}$$

with probability 1, which concludes the proof. □

C.3 Proof of Theorem 3.2

In this section, we provide the complete proof of Theorem 3.2. For convenience, we restate the theorem below.

Theorem C.3 (Theorem 3.2, averaging convergence rate)**.** *Consider the modification of Moshpit All-Reduce that works as follows: at each iteration $k \geq 1$ 1) peers are randomly split into r disjoint groups of sizes $M^k_1, \ldots, M^k_r$ in such a way that $\sum_{i=1}^r M^k_i = N$ and $M^k_i \geq 1$ $\forall i = 1, \ldots, r$ and 2) peers from each group compute their group average via All-Reduce. Let $\theta_1, \ldots, \theta_N$ be the input vectors of this procedure and $\theta^T_1, \ldots, \theta^T_N$ be the outputs after T iterations. Then,*

$$\mathbb{E}\left[\frac{1}{N}\sum_{i=1}^N \|\theta^T_i - \overline{\theta}\|^2\right] = \left(\frac{r-1}{N} + \frac{r}{N^2}\right)^T \cdot \frac{1}{N}\sum_{i=1}^N \|\theta_i - \overline{\theta}\|^2, \tag{18}$$

where $\overline{\theta} = \frac{1}{N}\sum_{i=1}^N \theta_i$.

Proof. First of all, let us clarify the procedure of random splitting of peers in r groups. We assume that at iteration k of the modified algorithm we generate a random permutation $\pi^k = (\pi_1^k, \ldots, \pi_N^k)$ of $1, \ldots, N$. Next, $J_1^k = \{\pi_1^k, \ldots, \pi_{M_1^k}^k\}$ form the indices of the first group of workers, $J_2^k = \{\pi_{M_1^k+1}^k, \ldots, \pi_{M_2^k}^k\}$ are the indices of the second group, and $J_r^k = \{\pi_{M_1^k+M_2^k+\ldots+M_{r-1}^k+1}^k, \ldots, \pi_N^k\}$ are the indices of group r. In other words, we generate a random permutation and take contiguous subgroups of indices corresponding to predefined group sizes M_i^k, starting from the first group.

By definition, we have $\bigsqcup_{i=1}^r J_i^k = \{1, 2, \ldots, N\}$, where $\sqcup$ defines the disjoint union operator. Moreover, notice that group sizes $M_1^k, \ldots, M_r^k$ can depend on k and even be random: for our analysis, it is sufficient that the randomness defining the permutation is independent from $M_1^k, \ldots, M_r^k$. Next, vectors $\theta_1^k, \ldots, \theta_N^k$ are obtained by the following formula:

$$\forall j = 1, \ldots, N, \quad \theta_j^k = \frac{1}{M_i^k} \sum_{t \in J_i^k} \theta_t^{k-1}, \quad \text{where } J_i^k \text{ is the group for which } j \in J_i^k.$$

Using this, we show that the average of vectors $\{\theta_i^k\}_{i=1}^n$ remains the same throughout the iterations of Moshpit All-Reduce:

$$\frac{1}{N}\sum_{j=1}^N \theta_j^k = \frac{1}{N}\sum_{i=1}^r M_i^k \cdot \frac{1}{M_i^k}\sum_{t \in J_i^k} \theta_t^{k-1} = \frac{1}{N}\sum_{i=1}^r \sum_{t \in J_i^k} \theta_t^{k-1} = \frac{1}{N}\sum_{j=1}^N \theta_j^{k-1}.$$

Therefore, the quantity $\frac{1}{N}\sum_{j=1}^N \|\theta_j^k - \overline{\theta}\|^2$ (average distortion) measures the quality of averaging. For this quantity, we can derive the following expression:

$$\begin{aligned}
\frac{1}{N}\sum_{j=1}^N \|\theta_j^k - \overline{\theta}\|^2 &= \frac{1}{N}\sum_{i=1}^r M_i^k \left\| \frac{1}{M_i^k}\sum_{t \in J_i^k} \theta_t^{k-1} - \overline{\theta} \right\|^2 \\
&= \frac{1}{N}\sum_{i=1}^r \frac{1}{M_i^k}\left(\sum_{t \in J_i^k} \|\theta_t^{k-1} - \overline{\theta}\|^2 + 2\sum_{t,l \in J_i^k, t<l} \langle \theta_t^{k-1} - \overline{\theta}, \theta_l^{k-1} - \overline{\theta} \rangle \right).
\end{aligned}$$

Taking the expectation $\mathbb{E}_{\pi^k}[\cdot]$ with respect to the randomness coming from the choice of π^k we get

$$\begin{aligned}
&\mathbb{E}_{\pi^k}\left[\frac{1}{N}\sum_{j=1}^N \|\theta_j^k - \overline{\theta}\|^2 \right] \\
&\qquad = \tfrac{1}{N}\sum_{i=1}^r \tfrac{1}{M_i^k}\left(\mathbb{E}_{\pi^k}\left[\sum_{t \in J_i^k} \|\theta_t^{k-1} - \overline{\theta}\|^2 \right] + 2\mathbb{E}_{\pi^k}\left[\sum_{t,l \in J_i^k, t<l} \langle \theta_t^{k-1} - \overline{\theta}, \theta_l^{k-1} - \overline{\theta} \rangle \right] \right).
\end{aligned}$$

Since $\forall j, j_1, j_2 \in \{1, \ldots, N\}, j_1 \neq j_2$ and for all $i = 1, \ldots, r$

$$\mathbb{P}\left\{ j \in J_i^k \right\} = \frac{M_i^k}{N}, \quad \mathbb{P}\left\{ j_1, j_2 \in J_i^k \right\} = \frac{M_i^k(M_i^k - 1)}{N^2},$$

we have

$$\begin{aligned}
\mathbb{E}_{\pi^k}\left[\frac{1}{N}\sum_{j=1}^N\|\theta_j^k - \overline{\theta}\|^2\right] &= \frac{1}{N}\sum_{i=1}^r\frac{1}{N}\sum_{j=1}^N\|\theta_j^{k-1}-\overline{\theta}\|^2 \\
&\quad+\frac{1}{N}\sum_{i=1}^r 2\frac{M_i^k-1}{N^2}\sum_{1\le j_1<j_2\le N}\langle\theta_{j_1}^{k-1}-\overline{\theta},\theta_{j_2}^{k-1}-\overline{\theta}\rangle \\
&= \frac{r}{N^2}\sum_{j=1}^N\|\theta_j^{k-1}-\overline{\theta}\|^2 + 2\frac{N-r}{N^3}\sum_{1\le j_1<j_2\le N}\langle\theta_{j_1}^{k-1}-\overline{\theta},\theta_{j_2}^{k-1}-\overline{\theta}\rangle \\
&= \left(\frac{r}{N^2}-\frac{N-r}{N^3}\right)\sum_{j=1}^N\|\theta_j^{k-1}-\overline{\theta}\|^2 + \frac{N-r}{N^3}\sum_{j=1}^N\|\theta_j^{k-1}-\overline{\theta}\|^2 \\
&\quad+2\frac{N-r}{N^3}\sum_{1\le j_1<j_2\le N}\langle\theta_{j_1}^{k-1}-\overline{\theta},\theta_{j_2}^{k-1}-\overline{\theta}\rangle \\
&= \frac{N(r-1)+r}{N^3}\sum_{j=1}^N\|\theta_j^{k-1}-\overline{\theta}\|^2 + \frac{N-r}{N^3}\underbrace{\left\|\sum_{j=1}^N(\theta_j^{k-1}-\overline{\theta})\right\|^2}_{\|N\overline{\theta}-N\overline{\theta}\|^2=0} \\
&= \left(\frac{r-1}{N}+\frac{r}{N^2}\right)\cdot\frac{1}{N}\sum_{j=1}^N\|\theta_j^{k-1}-\overline{\theta}\|^2.
\end{aligned}$$

Finally, we take the full expectation from the both sides of the above equation and apply the tower property $\mathbb{E}\left[\mathbb{E}_{\pi^k}\left[\cdot\right]\right] = \mathbb{E}\left[\cdot\right]$:

$$\mathbb{E}\left[\frac{1}{N}\sum_{j=1}^N\|\theta_j^k-\overline{\theta}\|^2\right] = \left(\frac{r-1}{N}+\frac{r}{N^2}\right)\mathbb{E}\left[\frac{1}{N}\sum_{j=1}^N\|\theta_j^{k-1}-\overline{\theta}\|^2\right].$$

Unrolling the recurrence for $k = T$, we establish (18). □

Remark C.1. *The result implies that increasing the group size $\alpha > 1$ times implies almost α times faster convergence to the average.*

Remark C.2. *Our analysis can be easily generalized to the case when number of groups r can depend on k and be a random variable independent from the choice of permutations and the number of groups at previous steps. In this case,* (18) *transforms into*

$$\mathbb{E}\left[\frac{1}{N}\sum_{i=1}^N\|\theta_i^T-\overline{\theta}\|^2\right] = \frac{1}{N}\sum_{i=1}^N\|\theta_i-\overline{\theta}\|^2\cdot\prod_{k=1}^T\left(\frac{\mathbb{E}[r_k]-1}{N}+\frac{\mathbb{E}[r_k]}{N^2}\right), \tag{19}$$

where r_k is the number of groups at iteration k.

C.4 Additional Guarantees For Moshpit Averaging

In this section, we derive the result measuring the rate of variance reduction when averaging random vectors with Algorithm 1. We start with the following technical lemma:

Lemma C.1. *Let $\xi \sim$ Binom(M,p) have a binomial distribution with parameters M (number of trials) and p (probability of success for each trial). Then*

$$\begin{aligned}
m_1(M,p) := \mathbb{E}\left[\min\left\{\frac{1}{\xi},1\right\}\right] &= (1-p)^M + \sum_{i=1}^M\frac{(1-p)^{M-i}-(1-p)^M}{i}, && (20)\\
m_2(M,p) := \mathbb{E}\left[\min\left\{\frac{1}{\xi^2},1\right\}\right] &= (1-p)^M + \sum_{i=1}^M\frac{(1-p)^{M-i}-(1-p)^M}{i}\sum_{j=i}^M\frac{1}{j}. && (21)
\end{aligned}$$

Proof. We start with the proof of (20). By definition of the expectation, we have

$$\mathbb{E}\left[\min\left\{\frac{1}{\xi}, 1\right\}\right] \quad = \quad (1-p)^M + \sum_{i=1}^{M} \frac{1}{i} p^i (1-p)^{M-i} \binom{M}{i}.$$

For simplicity of further derivations, we introduce the following notation: $m_1(M,p) = \mathbb{E}\left[\min\left\{\frac{1}{\xi}, 1\right\}\right]$ and $m_2(M,p) = \mathbb{E}\left[\min\left\{\frac{1}{\xi^2}, 1\right\}\right]$. Taking the derivative of $m_1(M,p)$ by p, we obtain

$$\begin{aligned}
m_1'(M,p) &= -M(1-p)^{M-1} + \sum_{i=1}^{M} p^{i-1}(1-p)^{M-i}\binom{M}{i} \\
&\quad - \sum_{i=1}^{M} \frac{M-i}{i} p^i (1-p)^{M-i-1} \binom{M}{i} \\
&= -M(1-p)^{M-1} + \frac{1}{p}\left(-(1-p)^M + \sum_{i=0}^{M} p^i (1-p)^{M-i}\binom{M}{i}\right) \\
&\quad - \frac{M}{1-p} \sum_{i=1}^{M} \frac{1}{i} p^i (1-p)^{M-i} \binom{M}{i} \\
&\quad + \frac{1}{1-p}\left(-(1-p)^M + \sum_{i=0}^{M} p^i (1-p)^{M-i}\binom{M}{i}\right) \\
&= -M(1-p)^{M-1} + \frac{1}{p}\left(1-(1-p)^M\right) - \frac{M}{1-p}\left(m_1(M,p) - (1-p)^M\right) \\
&\quad + \frac{1}{1-p}\left(1-(1-p)^M\right) \\
&= \frac{1}{p(1-p)} - \frac{(1-p)^{M-1}}{p} - \frac{M}{1-p} m_1(M,p).
\end{aligned}$$

Rearranging the terms, we get the following linear first-order ODE

$$m_1'(M,p) + \frac{M}{1-p} m_1(M,p) = \frac{1}{p(1-p)} - \frac{(1-p)^{M-1}}{p}. \tag{22}$$

To solve it, we consider the following homogeneous ODE:

$$m_1'(M,p) + \frac{M}{1-p} m_1(M,p) = 0.$$

The solution of this ODE is $m_1(M,p) = C(1-p)^M$, where $C \in \mathbb{R}$ is an arbitrary real constant. Next, we go back to the initial ODE (22) and try to find a solution of the form $m_1(M,p) = C(p)(1-p)^M$, where $C(p) : \mathbb{R} \to \mathbb{R}$ is a differentiable function:

$$\begin{aligned}
\left(C(p)(1-p)^M\right)' + \frac{M}{1-p} C(p)(1-p)^M &= \frac{1}{p(1-p)} - \frac{(1-p)^{M-1}}{p} \\
&\Downarrow \\
C'(p)(1-p)^M &= \frac{1}{p(1-p)} - \frac{(1-p)^{M-1}}{p} \\
&\Downarrow \\
C'(p) &= \frac{1}{p(1-p)^{M+1}} - \frac{1}{p(1-p)}.
\end{aligned}$$

Since

$$\frac{1}{x(1-x)^{k+1}} = \frac{1}{x(1-x)^k} + \frac{1}{(1-x)^{k+1}} \tag{23}$$

for all $x \notin \{0,1\}$ and all non-negative integers k, we have

$$\begin{array}{rcl} C'(p) & = & \frac{1}{p}+\frac{1}{1-p}+\frac{1}{(1-p)^2}+\ldots+\frac{1}{(1-p)^{M+1}}-\frac{1}{p}-\frac{1}{1-p} \\ & \Downarrow & \\ C'(p) & = & \sum_{i=1}^{M}(1-p)^{-i-1}, \end{array}$$

hence

$$C(p)=\hat{C}+\sum_{i=1}^{M}\frac{1}{i}(1-p)^{-i},$$

where $\hat{C}$ is a real constant. Putting all together, we obtain

$$m_1(M,p) \quad = \quad C(p)(1-p)^M=\hat{C}(1-p)^M+\sum_{i=1}^{M}\frac{1}{i}(1-p)^{M-i}.$$

Taking $m_1(M,0)=1$ into account, we conclude that $\hat{C}=1-\sum_{i=1}^{M}\frac{1}{i}$ and obtain (20).

Using a similar technique, we derive (21). By definition of the expectation, we have

$$m_2(M,p) \quad = \quad (1-p)^M+\sum_{i=1}^{M}\frac{1}{i^2}p^i(1-p)^{M-i}\binom{M}{i}.$$

Taking the derivative of $m_2(M,p)$ by p, we obtain

$$\begin{array}{rcl} m_2'(M,p) & = & -M(1-p)^{M-1}+\sum_{i=1}^{M}\frac{1}{i}p^{i-1}(1-p)^{M-i}\binom{M}{i} \\ & & -\sum_{i=1}^{M}\frac{M-i}{i^2}p^i(1-p)^{M-i-1}\binom{M}{i} \\ & = & -M(1-p)^{M-1}+\frac{1}{p}\sum_{i=1}^{M}\frac{1}{i}p^i(1-p)^{M-i}\binom{M}{i} \\ & & -\frac{M}{1-p}\sum_{i=1}^{M}\frac{1}{i^2}p^i(1-p)^{M-i}\binom{M}{i}+\frac{1}{1-p}\sum_{i=1}^{M}\frac{1}{i}p^i(1-p)^{M-i}\binom{M}{i} \\ & = & -M(1-p)^{M-1}+\frac{1}{p}\left(m_1(M,p)-(1-p)^M\right) \\ & & +\frac{1}{1-p}\left(-Mm_2(M,p)+M(1-p)^M+m_1(M,p)-(1-p)^M\right) \\ & = & \frac{m_1(M,p)}{p(1-p)}-\frac{(1-p)^{M-1}}{p}-\frac{M}{1-p}m_2(M,p). \end{array}$$

Rearranging the terms, we get the following linear first-order ODE

$$m_2'(M,p)+\frac{M}{1-p}m_2(M,p)=\frac{m_1(M,p)}{p(1-p)}-\frac{(1-p)^{M-1}}{p}. \tag{24}$$

To solve this ODE, we consider the homogeneous ODE:

$$m_2'(M,p)+\frac{M}{1-p}m_2(M,p)=0.$$

The solution of this ODE is $m_2(M,p)=C(1-p)^M$, where $C\in\mathbb{R}$ is an arbitrary real constant. Next, we go back to the initial ODE (24) and try to find a solution of the form $m_2(M,p)=C(p)(1-p)^M$, where $C(p) : \mathbb{R} \to \mathbb{R}$ is a differentiable function:

$$\begin{aligned}
\left(C(p)(1-p)^M\right)' + \frac{M}{1-p}C(p)(1-p)^M &= \frac{m_1(M,p)}{p(1-p)} - \frac{(1-p)^{M-1}}{p} \\
&\Downarrow \\
C'(p)(1-p)^M &= \frac{m_1(M,p)}{p(1-p)} - \frac{(1-p)^{M-1}}{p} \\
&\Downarrow \\
C'(p) &= \frac{m_1(M,p)}{p(1-p)^{M+1}} - \frac{1}{p(1-p)}.
\end{aligned}$$

Using (23) and (20), we derive

$$\begin{aligned}
C'(p) &\overset{(20)}{=} -\frac{\sum\limits_{i=1}^{M} \frac{1}{i}}{p(1-p)} + \frac{\sum\limits_{i=1}^{M} \frac{1}{i}(1-p)^{M-i}}{p(1-p)^{M+1}} \\
&= -\sum_{i=1}^{M} \frac{1}{ip(1-p)} + \sum_{i=1}^{M} \frac{1}{ip(1-p)^{i+1}} \\
&\overset{(23)}{=} -\sum_{i=1}^{M} \frac{1}{i}\left(\frac{1}{p} + \frac{1}{1-p}\right) \\
&\quad + \sum_{i=1}^{M} \frac{1}{i}\left(\frac{1}{p} + \frac{1}{1-p} + \frac{1}{(1-p)^2} + \ldots + \frac{1}{(1-p)^{i+1}}\right) \\
&= \sum_{i=1}^{M} \frac{1}{i}\left(\frac{1}{(1-p)^2} + \ldots + \frac{1}{(1-p)^{i+1}}\right) = \sum_{i=1}^{M} \frac{1}{(1-p)^{i+1}} \sum_{j=i}^{M} \frac{1}{j},
\end{aligned}$$

hence

$$C(p) = \hat{C} + \sum_{i=1}^{M} \frac{1}{i}(1-p)^{-i} \sum_{j=i}^{M} \frac{1}{j},$$

where $\hat{C}$ is a real constant. Putting all together, we obtain

$$m_2(M,p) \quad = \quad C(p)(1-p)^M = \hat{C}(1-p)^M + \sum_{i=1}^{M} \frac{1}{i}(1-p)^{M-i} \sum_{j=i}^{M} \frac{1}{j}.$$

Taking $m_2(M,0) = 1$ into account, we conclude that $\hat{C} = 1 - \sum_{i=1}^{M} \frac{1}{i} \sum_{j=i}^{M} \frac{1}{j}$ and obtain (21). □

Using this lemma, we derive the following result:

Theorem C.4. *Assume that peers participating in Moshpit Averaging have independent random vectors $\theta_1, \ldots, \theta_N$ with means $\overline{\theta}_1, \ldots, \overline{\theta}_N$ and variances bounded by σ^2 before the averaging. Let $\theta_1^T, \ldots, \theta_N^T$ be the outputs of Moshpit Averaging after T iterations. Finally, we assume that each peer from the grid can be dropped out for the whole averaging process before averaging independently from other peers, i.e., $N \sim$ Binom(M^d, p). Then, for all $i = 1, \ldots, N$ we have*

$$\mathbb{E}\left[\left\|\theta_i^T - \mathbb{E}_\theta\left[\theta_i^T\right]\right\|^2\right] \le M^{T-1}\sigma^2 m_1(M-1,p)\left(m_2(M-1,p)\right)^{T-1}, \tag{25}$$

where functions $m_1(M,p)$ and $m_2(M,p)$ are defined in (20) *and* (21) *respectively, and $\mathbb{E}_\theta\left[\cdot\right]$ denotes the expectation w.r.t. the randomness from $\theta_1, \ldots, \theta_N$. Moreover, if $p \ge \frac{2}{3}$ and $M \ge 11$, then $m_1(M-1,p) \le \frac{2}{M}$, $m_2(M-1,p) \le \frac{3}{M^2}$ and*

$$\mathbb{E}\left[\left\|\theta_i^T - \mathbb{E}_\theta\left[\theta_i^T\right]\right\|^2\right] \le \frac{2\sigma^2}{M(M/3)^{T-1}}. \tag{26}$$

Proof. First of all, we recall an equivalent formulation of Moshpit Averaging. Consider a hypercube $\{1, \ldots, M\}^d$. One can consider the elements of this hypercube as hyperindices and assign a unique hyperindex to each peer so that peers can be viewed as vertices in the hypercube. Then, during the k-th iteration of Moshpit All-Reduce, each worker computes the average among those peers that have hyperindices with the same values except the k-th index; in other words, peers compute averages along the k-th dimension of the hypercube. Next, if $N = 0$, we assume that $\theta_i^T = \mathbb{E}_\theta\left[\theta_i^T\right]$ and (25) holds for free. Therefore, to derive (25), we assume that $N > 0$.

More formally, we use the following notation: $\theta_{C_i} = \theta_i$ for all $i = 1, \ldots, N$, where $C_i = (c_1^i, c_2^i, \ldots, c_d^i)$, $c_j^i \in \{1, \ldots, M\}$ for all $j = 1, \ldots, M$, and $C_i \neq C_k$ for $i \neq k$. Let $\mathcal{C}$ be the set of hyperindices corresponding to all peers. Next, we use $\theta_{C_i}^t$ to define the vector stored on i-th peer after t iterations of Moshpit Averaging. Then, for all $i = 1, \ldots, N$ we have $\theta_{C_i}^0 = \theta_{C_i}$ and for all $t = 1, \ldots, d$

$$\theta_{C_i}^t = \frac{1}{b_{i,t}} \sum_{k \in J_{i,t}} \theta_{C_k}^{t-1},$$

where $J_{i,t} = \{k \in N \mid C_k = (c_1^k, \ldots, c_d^k) \in \mathcal{C} \text{ and } c_j^k = c_j^i \ \forall j \neq t\}$ and $b_{i,t} = |J_{i,t}|$. Using this, we derive the following formula for $\theta_{C_i}^t$:

$$\theta_i^T \equiv \theta_{C_i}^T = \frac{1}{b_{i,T}} \sum_{i_1 \in J_{i,T}} \frac{1}{b_{i_1,T-1}} \sum_{i_2 \in J_{i_1,T-1}} \frac{1}{b_{i_2,T-2}} \sum_{i_3 \in J_{i_2,T-1}} \cdots \frac{1}{b_{i_{T-1},1}} \sum_{i_T \in J_{i_{T-1},1}} \theta_{i_T}.$$

Taking the expectation w.r.t. $\theta_1, \ldots, \theta_N$, we get

$$\mathbb{E}_\theta\left[\theta_i^T\right] = \frac{1}{b_{i,T}} \sum_{i_1 \in J_{i,T}} \frac{1}{b_{i_1,T-1}} \sum_{i_2 \in J_{i_1,T-1}} \frac{1}{b_{i_2,T-2}} \sum_{i_3 \in J_{i_2,T-1}} \cdots \frac{1}{b_{i_{T-1},1}} \sum_{i_T \in J_{i_{T-1},1}} \overline{\theta}_{i_T}.$$

Using the independence of $\theta_1, \ldots, \theta_N$, we derive

$$\begin{aligned}
\mathbb{E}_\theta\left[\left\|\theta_i^T - \mathbb{E}_\theta\left[\theta_i^T\right]\right\|^2\right] &= \mathbb{E}_\theta\left[\left\|\sum_{i_1 \in J_{i,T}} \sum_{i_2 \in J_{i_1,T-1}} \cdots \sum_{i_T \in J_{i_{T-1},1}} \frac{\theta_{i_T} - \overline{\theta}_{i_T}}{b_{i,T} b_{i_1,T-1} \ldots b_{i_{T-1},1}}\right\|^2\right] \\
&= \sum_{i_1 \in J_{i,T}} \sum_{i_2 \in J_{i_1,T-1}} \cdots \sum_{i_T \in J_{i_{T-1},1}} \frac{\mathbb{E}_\theta\left[\|\theta_{i_T} - \overline{\theta}_{i_T}\|^2\right]}{b_{i,T}^2 b_{i_1,T-1}^2 \ldots b_{i_{T-1},1}^2} \\
&\leq \sum_{i_1 \in J_{i,T}} \sum_{i_2 \in J_{i_1,T-1}} \cdots \sum_{i_T \in J_{i_{T-1},1}} \frac{\sigma^2}{b_{i,T}^2 b_{i_1,T-1}^2 \ldots b_{i_{T-1},1}^2} \\
&= \sum_{i_1 \in J_{i,T}} \sum_{i_2 \in J_{i_1,T-1}} \cdots \sum_{i_{T-1} \in J_{i_{T-2},2}} \frac{\sigma^2}{b_{i,T}^2 b_{i_1,T-1}^2 \ldots b_{i_{T-2},2}^2 b_{i_{T-1},1}}.
\end{aligned}$$

Next, taking the full expectation from the both sides of the previous inequality and using the tower property, we obtain

$$\mathbb{E}\left[\left\|\theta_i^T - \mathbb{E}_\theta\left[\theta_i^T\right]\right\|^2\right] \leq \mathbb{E}\left[\sum_{i_1 \in J_{i,T}} \sum_{i_2 \in J_{i_1,T-1}} \cdots \sum_{i_{T-1} \in J_{i_{T-2},2}} \frac{\sigma^2}{b_{i,T}^2 b_{i_1,T-1}^2 \ldots b_{i_{T-2},2}^2 b_{i_{T-1},1}}\right]. \tag{27}$$

Notice that $J_{i_k,T-k} \cap J_{i_{k+1},T-k-1} = \{i_{k+1}\}$ for all $k = 0, \ldots, T-1$, where $i_0 = i$. Moreover, for $k_1, k_2 \in \{0, 1, \ldots, T\}$, $k_1 < k_2$ either $J_{i_{k_1},T-k_1} \cap J_{i_{k_2},T-k_2} = \{k_2\}$ or $J_{i_{k_1},T-k_1} \cap J_{i_{k_2},T-k_2} = \varnothing$. The first situation is possible iff $i_{k_1} = i_{k_1+1} = \ldots i_{k_2-1}$.

Taking these observations about sets $J_{i_k,T-k}$ into account, we consider the sets $J'_{i_k,T-k} = J_{i_k,T-k} \setminus \{i_k\}$ for $k = 0, 1, \ldots, T-1$. These sets are pairwise disjoint and their cardinalities $b'_{i_k,T-k} = |J'_{i_k,T-k}|$ satisfy the following relations: $b_{i_k,T-k} = 1 + b'_{i_k,T-k} \geq \max\{1, b'_{i_k,T-k}\} =: \hat{b}_{i_k,T-k}$ for $k = 1, 2, \ldots, T-1$. Moreover, $b'_{i,T}, b'_{i_1,T-1}, \ldots, b'_{i_{T-1},1}$ are independent random variables from the binomial distribution $\text{Binom}(M-1, p)$. Finally, we notice that the number of terms in (27) is upper-bounded by M^{T-1}, since $|J_{i,t}| \leq M$ for all $i = 1, \ldots, N$ and $t = 0, \ldots, T$.

Putting all together, we obtain

$$\begin{aligned}
\mathbb{E}\left[\left\|\theta_i^T-\mathbb{E}_\theta\left[\theta_i^T\right]\right\|^2\right] &\leq \mathbb{E}\left[\sum_{i_1\in J_{i,T}}\sum_{i_2\in J_{i_1,T-1}}\cdots\sum_{i_{T-1}\in J_{i_{T-2},2}}\frac{\sigma^2}{\hat{b}_{i,T}^2\hat{b}_{i_1,T-1}^2\ldots\hat{b}_{i_{T-2},2}^2\hat{b}_{i_{T-1},1}}\right]\\
&\leq M^{T-1}\sigma^2\mathbb{E}\left[\frac{1}{\hat{\xi}_1^2\hat{\xi}_2^2\ldots\hat{\xi}_{T-1}^2\hat{\xi}_T}\right]\\
&= M^{T-1}\sigma^2\mathbb{E}\left[\frac{1}{\hat{\xi}_1^2}\right]\mathbb{E}\left[\frac{1}{\hat{\xi}_2^2}\right]\ldots\mathbb{E}\left[\frac{1}{\hat{\xi}_{T-1}^2}\right]\mathbb{E}\left[\frac{1}{\hat{\xi}_T}\right],
\end{aligned}$$

where $\hat{\xi}_k^2=\max\{1,\xi_1^2\}$ for $k=1,\ldots,T$ and $\xi_1,\ldots,\xi_T$ are i.i.d. random variables having the binomial distribution Binom$(M-1,p)$. Then one can simplify the inequality above using Lemma C.1 and get

$$\mathbb{E}\left[\left\|\theta_i^T-\mathbb{E}_\theta\left[\theta_i^T\right]\right\|^2\right] \leq M^{T-1}\sigma^2 m_1(M-1,p)\left(m_2(M-1,p)\right)^{T-1},$$

where functions $m_1(M,p)$ and $m_2(M,p)$ are defined in (20) and (21) respectively.

Next, we simplify the obtained upper bound under the assumption that M and p are not too small; specifically, $M\geq 11$ and $p\geq 2/3$. From (20), we have

$$\begin{aligned}
m_1(M-1,p) &= (1-p)^{M-1}+\sum_{i=1}^{M-1}\frac{1}{i}\left((1-p)^{M-1-i}-(1-p)^{M-1}\right)\\
&\leq (1-p)^{M-1}\sum_{i=1}^{M-1}\frac{1}{i(1-p)^i}.
\end{aligned}$$

Since

$$\frac{1}{(k+1)(1-p)^{k+1}}\cdot\frac{k(1-p)^k}{1}=\frac{k}{(k+1)(1-p)}\xrightarrow[k\to\infty]{}\frac{1}{1-p}\geq 3,$$

we have

$$(1-p)^{M-1}\sum_{i=1}^{M-1}\frac{1}{i(1-p)^i}=\Theta\left((1-p)^M\cdot\frac{1}{M(1-p)^M}\right)=\Theta\left(\frac{1}{M}\right).$$

Using simple algebra, one can prove that for $M\geq 11$ and $p\geq 2/3$ the following inequality holds:

$$m_1(M-1,p)\leq(1-p)^{M-1}\sum_{i=1}^{M-1}\frac{1}{i(1-p)^i}\leq\frac{2}{M}.$$

Similarly, we analyze $m_2(M-1,p)$:

$$\begin{aligned}
m_2(M-1,p) &= (1-p)^{M-1}+\sum_{i=1}^{M-1}\frac{1}{i}\left((1-p)^{M-1-i}-(1-p)^{M-1}\right)\sum_{j=i}^{M-1}\frac{1}{j}\\
&\leq (1-p)^{M-1}\sum_{i=1}^{M-1}\frac{1}{i(1-p)^i}\sum_{j=i}^{M-1}\frac{1}{j}.
\end{aligned}$$

Since

$$\begin{aligned}
\frac{\frac{1}{k(1-p)^k}\sum\limits_{j=k}^{M-1}\frac{1}{j}}{\frac{1}{(k-1)(1-p)^{k-1}}\sum\limits_{j=k-1}^{M-1}\frac{1}{j}} &= \frac{(k-1)\sum\limits_{j=k}^{M-1}\frac{1}{j}}{k(1-p)\left(\frac{1}{k-1}+\sum\limits_{j=k}^{M-1}\frac{1}{j}\right)}\geq\frac{3(k-1)\cdot\frac{1}{k}}{k\left(\frac{1}{k-1}+\frac{1}{k}\right)}\\
&= \frac{3(k-1)^2}{k(2k-1)}\xrightarrow[k\to\infty]{}\frac{3}{2},
\end{aligned}$$

we have

$$(1-p)^{M-1}\sum_{i=1}^{M-1}\frac{1}{i(1-p)^i}\sum_{j=i}^{M-1}\frac{1}{j}=\Theta\left((1-p)^M\cdot\frac{1}{M^2(1-p)^M}\right)=\Theta\left(\frac{1}{M^2}\right).$$

Next, one can prove with simple algebra that for $M\geq 11$ and $p\geq 2/3$ the following inequality holds:

$$m_2(M-1,p)\leq(1-p)^{M-1}\sum_{i=1}^{M-1}\frac{1}{i(1-p)^i}\sum_{j=i}^{M-1}\frac{1}{j}\leq\frac{3}{M^2}.$$

Plugging the obtained upper bounds for $m_1(M-1,p)$ and $m_2(M-1,p)$ in (25), we obtain (26). □

D Convergence Proofs of Moshpit SGD

In this section, we provide the complete statements of the theorems establishing the convergence of Moshpit SGD together with the full proofs. First, we introduce all necessary definitions, basic inequalities and auxiliary lemmas; then we prove the convergence in strongly convex and convex cases; lastly, we provide the proofs for the non-convex case.

D.1 Definitions, Basic Facts and Auxiliary Results

Below we provide several classical definitions and results which are used in our proofs.

D.1.1 Standard Definitions from Optimization Theory

Definition D.1 (L-smoothness)**.** *A function $f:\mathbb{R}^n\to\mathbb{R}$ is called L-smooth if for all $x,y\in\mathbb{R}^n$, the following inequality holds:*

$$\|\nabla f(x)-\nabla f(y)\|\leq L\|x-y\|. \tag{28}$$

If the function f is L-smooth, then for all $x,y\in\mathbb{R}^n$

$$f(y)\leq f(x)+\langle\nabla f(x),y-x\rangle+\frac{L}{2}\|y-x\|^2. \tag{29}$$

Next, if f is additionally convex and x^* is its minimizer, then for all $x\in\mathbb{R}^d$

$$\|\nabla f(x)\|^2\leq 2L\left(f(x)-f(x^*)\right). \tag{30}$$

Definition D.2 (μ-strong convexity)**.** *A differentiable function $f:\mathbb{R}^n\to\mathbb{R}$ is called μ-strongly convex if there exists a constant $\mu\geq 0$ such that for all $x,y\in\mathbb{R}^n$*

$$f(y)\geq f(x)+\langle\nabla f(x),y-x\rangle+\frac{\mu}{2}\|y-x\|^2. \tag{31}$$

D.1.2 Basic Facts

For all $a,b,\theta_1,\ldots,\theta_N\in\mathbb{R}^n$ and $\alpha>0$, the following inequalities hold:

$$\|a+b\|^2 \leq 2\|a\|^2+2\|b\|^2, \tag{32}$$

$$\left\|\frac{1}{N}\sum_{i=1}^N\theta_i\right\|^2 \leq \frac{1}{N}\sum_{i=1}^N\|\theta_i\|^2, \tag{33}$$

$$\langle a,b\rangle \leq \frac{\|a\|^2}{2\alpha}+\frac{\alpha\|b\|^2}{2}. \tag{34}$$

D.1.3 Properties of Expectation

Variance decomposition. For a random vector $\eta\in\mathbb{R}^d$ and any deterministic vector $x\in\mathbb{R}^d$, the variance satisfies

$$\mathbb{E}\left[\|\eta-\mathbb{E}\eta\|^2\right]=\mathbb{E}\left[\|\eta-x\|^2\right]-\|\mathbb{E}\eta-x\|^2 \tag{35}$$

Tower property of expectation. For any random variables $\xi,\eta\in\mathbb{R}^d$ we have

$$\mathbb{E}\left[\xi\right]=\mathbb{E}\left[\mathbb{E}\left[\xi\mid\eta\right]\right] \tag{36}$$

under the assumption that $\mathbb{E}[\xi]$ and $\mathbb{E}\left[\mathbb{E}\left[\xi\mid\eta\right]\right]$ are well-defined.

D.1.4 Auxiliary Results

For the readers' convenience, we list all auxiliary results that we use in our proofs below. The first result is classical and establishes that the gradient descent step is a contractive operator.

Lemma D.1 (Lemma 6 from [59]). *For any L-smooth and μ-strongly convex function $f : \mathbb{R}^n \to \mathbb{R}$, points $x, y \in \mathbb{R}^n$, and stepsize $\gamma \in (0, {}^1\!/\!{}_L]$, the following inequality holds:*

$$\|x - \gamma \nabla f(x) - y + \gamma \nabla f(y)\|^2 \leq (1 - \gamma\mu)\|x - y\|^2. \tag{37}$$

The next two lemmas are useful for estimating typical recurrences appearing in the analysis.

Lemma D.2 (Lemma I.2 from [60]). *Let $\{r_k\}_{k\geq 0}$ satisfy*

$$r_K \leq \frac{a}{\gamma W_K} + c_1\gamma + c_2\gamma^2$$

for all $K \geq 0$ with some constants $a, c_2 \geq 0$, $c_1 \geq 0$, where $w_k = (1 - \gamma\mu(1 - \delta_{pv,1}))^{-(k+1)}$, $W_K = \sum_{k=0}^K w_k$, $\mu > 0$, $\delta_{pv,1} \in [0, 1)$ and $\gamma \leq \gamma_0$ for some $\gamma_0 > 0$, $\gamma_0 \leq {}^1\!/\!{}_{\mu(1-\delta_{pv,1})}$. Then, for all K such that

$$\text{either } \frac{\ln\left(\max\left\{2, \min\left\{{}^{a\mu^2(1-\delta_{pv,1})^2K^2}\!/\!{}_{c_1}, {}^{a\mu^3(1-\delta_{pv,1})^3K^3}\!/\!{}_{c_2}\right\}\right\}\right)}{K} \leq 1$$

$$\text{or } \gamma_0 \leq \frac{\ln\left(\max\left\{2, \min\left\{{}^{a\mu^2(1-\delta_{pv,1})^2K^2}\!/\!{}_{c_1}, {}^{a\mu^3(1-\delta_{pv,1})^3K^3}\!/\!{}_{c_2}\right\}\right\}\right)}{(1-\delta_{pv,1})\mu K}$$

and

$$\gamma = \min\left\{\gamma_0, \frac{\ln\left(\max\left\{2, \min\left\{{}^{a\mu^2(1-\delta_{pv,1})^2K^2}\!/\!{}_{c_1}, {}^{a\mu^3(1-\delta_{pv,1})^3K^3}\!/\!{}_{c_2}\right\}\right\}\right)}{(1-\delta_{pv,1})\mu K}\right\}$$

we have that

$$r_K = \widetilde{\mathcal{O}}\left(\frac{a}{\gamma_0}\exp\left(-\gamma_0\mu(1-\delta_{pv,1})K\right) + \frac{c_1}{(1-\delta_{pv,1})\mu K} + \frac{c_2}{(1-\delta_{pv,1})^2\mu^2K^2}\right).$$

Lemma D.3 (Lemma I.3 from [60]). *Let $\{r_k\}_{k\geq 0}$ satisfy*

$$r_K \leq \frac{a}{\gamma K} + c_1\gamma + c_2\gamma^2$$

for all $K \geq 0$ with some constants $a, c_2 \geq 0$, $c_1 \geq 0$ where $\gamma \leq \gamma_0$ for some $\gamma_0 > 0$. Then for all K and

$$\gamma = \min\left\{\gamma_0, \sqrt{\frac{a}{c_1K}}, \sqrt[3]{\frac{a}{c_2K}}\right\}$$

we have that

$$r_K = \mathcal{O}\left(\frac{a}{\gamma_0 K} + \sqrt{\frac{ac_1}{K}} + \frac{\sqrt[3]{a^2c_2}}{K^{2/3}}\right).$$

Finally, the lemma below is useful for our convergence analysis in the non-convex case.

Lemma D.4 (Lemma I.1 from [60]). *For any τ random vectors $\xi_1, \ldots, \xi_\tau \in \mathbb{R}^d$ such that $\forall t = 2, \ldots, \tau$ the random vector ξ_t depends on $\xi_1, \ldots, \xi_{t-1}$ and does not depend on $\xi_{t+1}, \ldots, \xi_\tau$ the following inequality holds*

$$\mathbb{E}\left[\left\|\sum_{t=1}^{\tau}\xi_t\right\|^2\right] \leq e\tau\sum_{t=1}^{\tau}\mathbb{E}\left[\|\mathbb{E}_t[\xi_t]\|^2\right] + e\sum_{t=1}^{\tau}\mathbb{E}\left[\|\xi_t - \mathbb{E}_t[\xi_t]\|^2\right], \tag{38}$$

where $\mathbb{E}_t[\cdot]$ denotes the conditional expectation $\mathbb{E}[\,\cdot \mid \xi_{t-1}, \ldots, \xi_1]$.

D.2 Convex Case

In this section, we give the full proof of Theorem 3.3 about the convergence of Moshpit SGD for convex and strongly convex problems. The scheme of the proof follows the similar steps as in the state-of-the-art analysis of Local-SGD [61, 62, 60]. We start with the following lemma:

Lemma D.5. *Let $f_1 = \ldots = f_N = f$, function f be μ-strongly convex (Def. D.2) and L-smooth (see Def. D.1), and Assumptions 3.1 and 3.2 hold with $\Delta_{pv}^k = \delta_{pv,1}\gamma\mu\mathbb{E}[\|\theta^k - \theta^*\|^2] + \gamma^2\delta_{pv,2}^2$ and $\widetilde{\theta} = \theta^*$, where $\theta^* \in \operatorname{argmin}_{\theta\in\mathbb{R}^n} f(\theta)$ and $\delta_{pv,1} \in [0,1)$, $\delta_{pv,2} \geq 0$. Then, for any $k \geq 0$ the iterates produced by Moshpit SGD with $\gamma \leq {}^1\!/_{4L}$ satisfy*

$$\begin{aligned}\gamma\mathbb{E}\left[f(\theta^k) - f(\theta^*)\right] &\leq (1 - \gamma\mu(1-\delta_{pv,1}))\mathbb{E}\left[\|\theta^k - \theta^*\|^2\right] - \mathbb{E}\left[\|\theta^{k+1} - \theta^*\|^2\right] \\ &\quad + \frac{3L\gamma}{2}\mathbb{E}[V_k] + \gamma^2\left(\frac{\sigma^2}{N_{\min}} + \delta_{pv,2}^2\right), \end{aligned} \tag{39}$$

where $V_k = \frac{1}{N_k}\sum_{i\in P_k}\|\theta_i^k - \theta^k\|^2$ and $\theta^k = \frac{1}{N_k}\sum_{i\in P_k}\theta_i^k$.

Proof. Recall that Assumption 3.2 with $\Delta_{pv}^k = \delta_{pv,1}\gamma\mu\mathbb{E}[\|\theta^k - \theta^*\|^2] + \gamma^2\delta_{pv,2}^2$ and $\widetilde{\theta} = \theta^*$ states

$$\mathbb{E}\left[\langle\theta^{k+1} - \widehat{\theta}^{k+1}, \theta^{k+1} + \widehat{\theta}^{k+1} - 2\theta^*\rangle\right] \leq \delta_{pv,1}\gamma\mu\mathbb{E}[\|\theta^k - \theta^*\|^2] + \gamma^2\delta_{pv,2}^2, \tag{40}$$

where $\widehat{\theta}^{k+1} = \frac{1}{N_k}\sum_{i\in P_k}(\theta_i^k - \gamma g_i^k)$. Next, the definition of $\widehat{\theta}^{k+1}$ implies

$$\widehat{\theta}^{k+1} = \frac{1}{N_k}\sum_{i\in P_k}\theta_i^k - \frac{\gamma}{N_k}\sum_{i\in P_k}g_i^k = \theta^k - \gamma g^k,$$

where $g^k = \frac{1}{N_k}\sum_{i\in P_k} g_i^k$. Using this, we derive

$$\begin{aligned}\|\theta^{k+1} - \theta^*\|^2 &= \|\widehat{\theta}^{k+1} - \theta^*\|^2 + 2\langle\theta^{k+1} - \widehat{\theta}^{k+1}, \widehat{\theta}^{k+1} - \theta^*\rangle + \|\theta^{k+1} - \widehat{\theta}^{k+1}\|^2 \\ &= \|\theta^k - \theta^* - \gamma g^k\|^2 + \langle\theta^{k+1} - \widehat{\theta}^{k+1}, \theta^{k+1} + \widehat{\theta}^{k+1} - 2\theta^*\rangle \\ &= \|\theta^k - \theta^*\|^2 - 2\gamma\langle\theta^k - \theta^*, g^k\rangle + \gamma^2\|g^k\|^2 \\ &\quad + \langle\theta^{k+1} - \widehat{\theta}^{k+1}, \theta^{k+1} + \widehat{\theta}^{k+1} - 2\theta^*\rangle.\end{aligned}$$

Taking the conditional expectation $\mathbb{E}\left[\cdot \mid \theta^k\right] := \mathbb{E}\left[\cdot \mid P_k, \theta_i^k, i \in P_k\right]$ from the both sides of the previous equation and using Assumption 3.1, we obtain

$$\begin{aligned}\mathbb{E}\left[\|\theta^{k+1} - \theta^*\|^2 \mid \theta^k\right] &= \|\theta^k - \theta^*\|^2 - 2\gamma\left\langle\theta^k - \theta^*, \frac{1}{N_k}\sum_{i\in P_k}\nabla f(\theta_i^k)\right\rangle \\ &\quad + \gamma^2\mathbb{E}\left[\left\|\frac{1}{N_k}\sum_{i\in P_k}g_i^k\right\|^2 \mid \theta^k\right] \\ &\quad + \mathbb{E}\left[\langle\theta^{k+1} - \widehat{\theta}^{k+1}, \theta^{k+1} + \widehat{\theta}^{k+1} - 2\theta^*\rangle \mid \theta^k\right].\end{aligned} \tag{41}$$

Next, we estimate the second and the third terms in the right-hand side of (41). First,

$$\begin{aligned}-2\gamma\left\langle\theta^k - \theta^*, \frac{1}{N_k}\sum_{i\in P_k}\nabla f(\theta_i^k)\right\rangle &= \frac{2\gamma}{N_k}\sum_{i\in P_k}\left(\langle\theta^* - \theta_i^k, \nabla f(\theta_i^k)\rangle + \langle\theta_i^k - \theta^k, \nabla f(\theta_i^k)\rangle\right) \\ &\overset{(31),(29)}{\leq} \frac{2\gamma}{N_k}\sum_{i\in P_k}\left(f(\theta^*) - f(\theta_i^k) - \frac{\mu}{2}\|\theta_i^k - \theta^*\|^2\right) \\ &\quad + \frac{2\gamma}{N_k}\sum_{i\in P_k}\left(f(\theta_i^k) - f(\theta^k) + \frac{L}{2}\|\theta_i^k - \theta^k\|^2\right) \\ &\overset{(33)}{\leq} 2\gamma\left(f(\theta^*) - f(\theta^k)\right) - \gamma\mu\|\theta^k - \theta^*\|^2 + L\gamma V_k,\end{aligned} \tag{42}$$

where $V_k = \frac{1}{N_k}\sum_{i\in P_k}\|\theta_i^k - \theta^k\|^2$. Secondly, since stochastic gradients $\{g_i^k\}_{i\in P_k}$ are computed independently, we get

$$\begin{aligned}
\gamma^2\mathbb{E}\left[\left\|\frac{1}{N_k}\sum_{i\in P_k} g_i^k\right\|^2 \mid \theta^k\right] &\overset{(35)}{=} \gamma^2\left\|\frac{1}{N_k}\sum_{i\in P_k}\nabla f(\theta_i^k)\right\|^2 \\
&\quad +\gamma^2\mathbb{E}\left[\left\|\frac{1}{N_k}\sum_{i\in P_k}(g_i^k - \nabla f(\theta_i^k))\right\|^2 \mid \theta^k\right] \\
&\overset{(33)}{\leq} 2\gamma^2\left\|\frac{1}{N_k}\sum_{i\in P_k}(\nabla f(\theta_i^k) - \nabla f(\theta^k))\right\|^2 + 2\gamma^2\|\nabla f(\theta^k)\|^2 \\
&\quad +\frac{\gamma^2}{N_k^2}\sum_{i\in P_k}\mathbb{E}\left[\|g_i^k - \nabla f(\theta_i^k)\|^2 \mid \theta^k\right] \\
&\overset{(33),(30),(7)}{\leq} \frac{2\gamma^2}{N_k}\sum_{i\in P_k}\|\nabla f(\theta_i^k) - \nabla f(\theta^k)\|^2 \\
&\quad +4L\gamma^2\left(f(\theta^k) - f(\theta^*)\right) + \frac{\gamma^2\sigma^2}{N_k} \\
&\overset{(28)}{\leq} \underbrace{\frac{2L^2\gamma^2}{N_k}\sum_{i\in P_k}\|\theta_i^k - \theta^k\|^2}_{2L^2\gamma^2 V_k} \\
&\quad +4L\gamma^2\left(f(\theta^k) - f(\theta^*)\right) + \frac{\gamma^2\sigma^2}{N_{\min}}. \qquad (43)
\end{aligned}$$

Plugging (42) and (43) in (41), we obtain

$$\begin{aligned}
\mathbb{E}\left[\|\theta^{k+1} - \theta^*\|^2 \mid \theta^k\right] &\leq (1-\gamma\mu)\|\theta^k - \theta^*\|^2 - 2\gamma\left(1 - 2L\gamma\right)\left(f(\theta^k) - f(\theta^*)\right) \\
&\quad +L\gamma\left(1 + 2L\gamma\right)V_k + \frac{\gamma^2\sigma^2}{N_{\min}} \\
&\quad +\mathbb{E}\left[\langle\theta^{k+1} - \widehat{\theta}^{k+1}, \theta^{k+1} + \widehat{\theta}^{k+1} - 2\theta^*\rangle \mid \theta^k\right],
\end{aligned}$$

and

$$\begin{aligned}
\mathbb{E}\left[\|\theta^{k+1} - \theta^*\|^2\right] &\overset{(40)}{\leq} (1-\gamma\mu(1-\delta_{pv,1}))\mathbb{E}\left[\|\theta^k - \theta^*\|^2\right] - 2\gamma\left(1 - 2L\gamma\right)\mathbb{E}\left[f(\theta^k) - f(\theta^*)\right] \\
&\quad +L\gamma\left(1 + 2L\gamma\right)\mathbb{E}[V_k] + \gamma^2\left(\frac{\sigma^2}{N_{\min}} + \delta_{pv,2}^2\right) \\
&\leq (1-\gamma\mu(1-\delta_{pv,1}))\mathbb{E}\left[\|\theta^k - \theta^*\|^2\right] - \gamma\mathbb{E}\left[f(\theta^k) - f(\theta^*)\right] \\
&\quad +\frac{3L\gamma}{2}\mathbb{E}[V_k] + \gamma^2\left(\frac{\sigma^2}{N_{\min}} + \delta_{pv,2}^2\right),
\end{aligned}$$

where in the last inequality we use $\gamma \leq 1/4L$. □

Next, we estimate the term $\mathbb{E}[V_k]$ measuring the expected dissimilarity between local iterates and their global average at iteration k.

Lemma D.6. *Let $f_1 = \ldots = f_N = f$, function f be μ-strongly convex (Def. D.2) and L-smooth (see Def. D.1), and Assumptions 3.1 and 3.2 hold with $\Delta_{pv}^k = \delta_{pv,1}\gamma\mu\mathbb{E}[\|\theta^k - \theta^*\|^2] + \gamma^2\delta_{pv,2}^2$ and $\widetilde{\theta} = \theta^*$, where $\theta^* \in \operatorname{argmin}_{\theta\in\mathbb{R}^n} f(\theta)$ and $\delta_{pv,1} \in [0,1)$, $\delta_{pv,2} \geq 0$. Then, for any $k \geq 0$ the iterates produced by Moshpit SGD with $\gamma \leq 1/4L$ satisfy*

$$\mathbb{E}[V_k] \leq 2\gamma^2\left(4\delta_{aq}^2 + (\tau - 1)\sigma^2\right), \qquad (44)$$

where $V_k = \frac{1}{N_k}\sum_{i\in P_k}\|\theta_i^k - \theta^k\|^2$ and $\theta^k = \frac{1}{N_k}\sum_{i\in P_k}\theta_i^k$.

Proof. First of all, if $k = a\tau$ for some integer $a \geq 0$, then (44) follows from Assumption 3.2 (eq. (10)). Therefore, we consider such k that $k = a\tau + t'$ for some $t' \in (0, \tau)$. Then, for any $i, j \in P_k$, $i \neq j$

$$\begin{aligned}
\mathbb{E}\left[\|\theta_i^k - \theta_j^k\|^2 \mid \theta^{k-1}\right] &= \mathbb{E}\left[\|\theta_i^{k-1} - \gamma g_i^{k-1} - \theta_j^{k-1} + \gamma g_j^{k-1}\|^2 \mid \theta^{k-1}\right] \\
&\overset{(35)}{=} \|\theta_i^{k-1} - \gamma\nabla f(\theta_i^{k-1}) - \theta_j^{k-1} + \gamma\nabla f(\theta_j^{k-1})\|^2 \\
&\quad + \gamma^2\mathbb{E}\left[\|g_i^{k-1} - \nabla f(\theta_i^{k-1}) + g_j^{k-1} - \nabla f(\theta_j^{k-1})\|^2 \mid \theta^{k-1}\right].
\end{aligned}$$

Using Lemma D.1 and independence of g_i^{k-1} and g_j^{k-1} for given $\theta_i^{k-1}, \theta_j^{k-1}$, $i \neq j$ we derive

$$\begin{aligned}
\mathbb{E}\left[\|\theta_i^k - \theta_j^k\|^2 \mid \theta^{k-1}\right] &\overset{(37)}{\leq} (1 - \gamma\mu)\|\theta_i^{k-1} - \theta_j^{k-1}\|^2 + \gamma^2\mathbb{E}\left[\|g_i^{k-1} - \nabla f(\theta_i^{k-1})\|^2 \mid \theta^{k-1}\right] \\
&\quad + \gamma^2\mathbb{E}\left[\|g_j^{k-1} - \nabla f(\theta_j^{k-1})\|^2 \mid \theta^{k-1}\right] \\
&\overset{(7)}{\leq} (1 - \gamma\mu)\|\theta_i^{k-1} - \theta_j^{k-1}\|^2 + 2\gamma^2\sigma^2,
\end{aligned}$$

from which we get the following:

$$\mathbb{E}_g\left[\|\theta_i^k - \theta_j^k\|^2\right] \leq (1 - \gamma\mu)\mathbb{E}_g\left[\|\theta_i^{k-1} - \theta_j^{k-1}\|^2\right] + 2\gamma^2\sigma^2 \leq \mathbb{E}_g\left[\|\theta_i^{k-1} - \theta_j^{k-1}\|^2\right] + 2\gamma^2\sigma^2.$$

Here, $\mathbb{E}_g[\cdot]$ denotes the expectation conditioned on $\{P_k\}_{k=a\tau}^{(a+1)\tau-1}$. Unrolling the recurrence, we get

$$\begin{aligned}
\mathbb{E}_g\left[\|\theta_i^k - \theta_j^k\|^2\right] &\leq \mathbb{E}_g\left[\|\theta_i^{a\tau} - \theta_j^{a\tau}\|^2\right] + 2(k - a\tau)\gamma^2\sigma^2 \\
&\leq \mathbb{E}_g\left[\|\theta_i^{a\tau} - \theta_j^{a\tau}\|^2\right] + 2(\tau - 1)\gamma^2\sigma^2. \qquad (45)
\end{aligned}$$

Using this, we estimate $\mathbb{E}_g[V_k]$:

$$\begin{aligned}
\mathbb{E}_g[V_k] &= \frac{1}{N_k}\sum_{i\in P_k}\mathbb{E}_g\left[\left\|\theta_i^k - \frac{1}{N_k}\sum_{j\in P_k}\theta_j^k\right\|^2\right] \overset{(33)}{\leq} \frac{1}{N_k^2}\sum_{i,j\in P_k}\mathbb{E}_g\left[\|\theta_i^k - \theta_j^k\|^2\right] \\
&\overset{(45)}{\leq} \frac{1}{N_k^2}\sum_{i,j\in P_k}\mathbb{E}_g\left[\|\theta_i^{a\tau} - \theta_j^{a\tau}\|^2\right] + 2(\tau - 1)\gamma^2\sigma^2 \\
&\overset{(32)}{\leq} \frac{2}{N_k^2}\sum_{i,j\in P_k}\left(\mathbb{E}_g\left[\|\theta_i^{a\tau} - \theta^{a\tau}\|^2\right] + \mathbb{E}_g\left[\|\theta_j^{a\tau} - \theta^{a\tau}\|^2\right]\right) + 2(\tau - 1)\gamma^2\sigma^2 \\
&= \frac{4}{N_k}\sum_{i\in P_k}\mathbb{E}_g\left[\|\theta_i^{a\tau} - \theta^{a\tau}\|^2\right] + 2(\tau - 1)\gamma^2\sigma^2 \\
&\leq \frac{4}{N_{a\tau}}\cdot\frac{N_{a\tau}}{N_k}\sum_{i\in P_{a\tau}}\mathbb{E}_g\left[\|\theta_i^{a\tau} - \theta^{a\tau}\|^2\right] + 2(\tau - 1)\gamma^2\sigma^2 \\
&\leq \mathbb{E}_g\left[\frac{8}{N_{a\tau}}\sum_{i\in P_{a\tau}}\|\theta_i^{a\tau} - \theta^{a\tau}\|^2\right] + 2(\tau - 1)\gamma^2\sigma^2,
\end{aligned}$$

where in the last inequality we use $2N_{(a+1)\tau} = 2|P_{(a+1)\tau}| \geq |P_{a\tau}| = N_{a\tau}$ and $|N_k| \leq |N_{k-1}|$ following from Assumption 3.2. Finally, we take the full expectation from the previous inequality:

$$\mathbb{E}[V_k] \overset{(36)}{\leq} 8\mathbb{E}\left[\frac{1}{N_{a\tau}}\sum_{i\in P_{a\tau}}\|\theta_i^{a\tau} - \theta^{a\tau}\|^2\right] + 2(\tau - 1)\gamma^2\sigma^2 \overset{(10)}{\leq} 2\gamma^2\left(4\delta_{aq}^2 + (\tau - 1)\sigma^2\right).$$

This finishes the proof. □

Combining Lemmas D.5 and D.6, we get the following result:

Theorem D.1 (Theorem 3.3, convergence in the convex case)**.** *Let $f_1 = \ldots = f_N = f$ be μ-strongly convex (Def. D.2) and L-smooth (see Def. D.1), and Assumptions 3.1 and 3.2 hold with*

$\Delta_{pv}^k = \delta_{pv,1}\gamma\mu\mathbb{E}[\|\theta^k - \theta^*\|^2] + \gamma^2\delta_{pv,2}^2$ *and* $\widetilde{\theta} = \theta^*$, *where* $\theta^* \in \mathrm{argmin}_{\theta\in\mathbb{R}^n} f(\theta)$ *and* $\delta_{pv,1} \in [0,1)$, $\delta_{pv,2} \geq 0$. *Then, for any* $K \geq 0$, *the iterates produced by Moshpit SGD with* $\gamma \leq {}^1\!/_{4L}$ *satisfy*

$$\begin{aligned}\mathbb{E}\left[f(\overline{\theta}^K) - f(\theta^*)\right] \quad &\leq \quad (1-\gamma\mu(1-\delta_{pv,1}))^K\frac{R_0^2}{\gamma}\\ &\qquad + \gamma\left(\frac{\sigma^2}{N_{\min}} + \delta_{pv,2}^2 + 3L\gamma\left(4\delta_{aq}^2 + (\tau-1)\sigma^2\right)\right), \end{aligned} \tag{46}$$

when $\mu > 0$, *and*

$$\mathbb{E}\left[f(\overline{\theta}^K) - f(\theta^*)\right] \leq \frac{R_0^2}{\gamma K} + \gamma\left(\frac{\sigma^2}{N_{\min}} + \delta_{pv,2}^2 + 3L\gamma\left(4\delta_{aq}^2 + (\tau-1)\sigma^2\right)\right), \tag{47}$$

when $\mu = 0$, *where* $R_0 = \|\theta^0 - \theta^*\|$, $\overline{\theta}^K = \frac{1}{W_K}\sum_{k=0}^K w_k\theta^k = \frac{1}{W_K}\sum_{k=0}^K \frac{w_k}{N_k}\sum_{i\in P_k}\theta_i^k$, $w_k = (1 - \gamma\mu(1-\delta_{pv,1}))^{-(k+1)}$, *and* $W_K = \sum_{k=0}^K w_k$. *That is, Moshpit SGD achieves* $\mathbb{E}[f(\overline{\theta}^K) - f(\theta^*)] \leq \varepsilon$ *after*

$$K = \widetilde{\mathcal{O}}\left(\frac{L}{(1-\delta_{pv,1})\mu} + \frac{\sigma^2}{N_{\min}(1-\delta_{pv,1})\mu\varepsilon} + \frac{\delta_{pv,2}^2}{(1-\delta_{pv,1})\mu\varepsilon} + \sqrt{\frac{L((\tau-1)\sigma^2 + \delta_{aq}^2)}{(1-\delta_{pv,1})^2\mu^2\varepsilon}}\right) \tag{48}$$

iterations with

$$\gamma = \min\left\{\frac{1}{4L}, \frac{\ln\left(\max\left\{2, \min\left\{\frac{R_0^2\mu^2(1-\delta_{pv,1})^2K^2}{(\delta_{pv,2}^2 + {}^{\sigma^2}\!/_{N_{\min}})}, \frac{R_0^2\mu^3(1-\delta_{pv,1})^3K^3}{3L\left(4\delta_{aq}^2 + (\tau-1)\sigma^2\right)}\right\}\right\}\right)}{(1-\delta_{pv,1})\mu K}\right\}$$

when $\mu > 0$, *and after*

$$K = \mathcal{O}\left(\frac{LR_0^2}{\varepsilon} + \frac{R_0^2\sigma^2}{N_{\min}\varepsilon^2} + \frac{R_0^2\delta_{pv,2}^2}{\varepsilon^2} + \frac{R_0^2\sqrt{L((\tau-1)\sigma^2 + \delta_{aq}^2)}}{\varepsilon^{3/2}}\right) \tag{49}$$

iterations with

$$\gamma = \min\left\{\frac{1}{4L}\sqrt{\frac{R_0}{(\delta_{pv,2}^2 + {}^{\sigma^2}\!/_{N_{\min}})K}}, \sqrt[3]{\frac{R_0^2}{3L\left(4\delta_{aq}^2 + (\tau-1)\sigma^2\right)K}}\right\}$$

when $\mu = 0$.

Proof. Plugging the result of Lemma D.6 in inequality (39) from Lemma D.5, we obtain

$$\begin{aligned}\gamma\mathbb{E}\left[f(\theta^k) - f(\theta^*)\right] \quad &\leq \quad (1-\gamma\mu(1-\delta_{pv,1}))\mathbb{E}\left[\|\theta^k - \theta^*\|^2\right] - \mathbb{E}\left[\|\theta^{k+1} - \theta^*\|^2\right]\\ &\qquad + 3L\gamma^3\left(4\delta_{aq}^2 + (\tau-1)\sigma^2\right) + \gamma^2\left(\frac{\sigma^2}{N_{\min}} + \delta_{pv,2}^2\right).\end{aligned}$$

Next, we sum up these inequalities for $k = 0, \ldots, K$ with weights $w_k = (1 - \gamma\mu(1-\delta_{pv,1}))^{-(k+1)}$ and divide both sides by γW_K, where $W_K = \sum_{k=0}^K w_k$:

$$\begin{aligned}
\frac{1}{W_K}\sum_{k=0}^K w_k \mathbb{E}\left[f(\theta^k) - f(\theta^*)\right] &\leq \frac{1}{\gamma W_K}\sum_{k=0}^K (1-\gamma\mu(1-\delta_{pv,1}))w_k\mathbb{E}\left[\|\theta^k - \theta^*\|^2\right] \\
&\quad -\frac{1}{\gamma W_K}\sum_{k=0}^K w_k\mathbb{E}\left[\|\theta^{k+1} - \theta^*\|^2\right] \\
&\quad +\gamma\left(\frac{\sigma^2}{N_{\min}} + \delta_{pv,2}^2 + 3L\gamma\left(4\delta_{aq}^2 + (\tau-1)\sigma^2\right)\right) \\
&= \frac{1}{\gamma W_K}\sum_{k=0}^K \left(w_{k-1}\mathbb{E}\left[\|\theta^k - \theta^*\|^2\right] - w_k\mathbb{E}\left[\|\theta^{k+1} - \theta^*\|^2\right]\right) \\
&\quad +\gamma\left(\frac{\sigma^2}{N_{\min}} + \delta_{pv,2}^2 + 3L\gamma\left(4\delta_{aq}^2 + (\tau-1)\sigma^2\right)\right) \\
&= \frac{w_{-1}\|\theta^0 - \theta^*\|^2 - w_K\mathbb{E}\left[\|\theta^{K+1} - \theta^*\|^2\right]}{\gamma W_K} \\
&\quad +\gamma\left(\frac{\sigma^2}{N_{\min}} + \delta_{pv,2}^2 + 3L\gamma\left(4\delta_{aq}^2 + (\tau-1)\sigma^2\right)\right) \\
&\leq \frac{\|\theta^0 - \theta^*\|^2}{\gamma W_K} \\
&\quad +\gamma\left(\frac{\sigma^2}{N_{\min}} + \delta_{pv,2}^2 + 3L\gamma\left(4\delta_{aq}^2 + (\tau-1)\sigma^2\right)\right).
\end{aligned}$$

Since f is convex, we apply the Jensen's inquality

$$f\left(\frac{1}{W_K}\sum_{k=0}^K w_k\theta^k\right) \leq \frac{1}{W_K}\sum_{k=0}^K w_k f(\theta^k)$$

to the previous result and get

$$\mathbb{E}\left[f(\overline{\theta}^K) - f(\theta^*)\right] \leq \frac{R_0^2}{\gamma W_K} + \gamma\left(\frac{\sigma^2}{N_{\min}} + \delta_{pv,2}^2 + 3L\gamma\left(4\delta_{aq}^2 + (\tau-1)\sigma^2\right)\right),$$

where $R_0 = \|\theta^0 - \theta^*\|$ and $\overline{\theta}^K = \frac{1}{W_K}\sum_{k=0}^K w_k\theta^k = \frac{1}{W_K}\sum_{k=0}^K \frac{w_k}{N_k}\sum_{i\in P_k}\theta_i^k$. If $\mu > 0$, then $W_K \geq w_K \geq (1-\gamma\mu(1-\delta_{pv,1}))^{-K}$, implying (46). Next, $w_k = 1$ and $W_K = K$ when $\mu = 0$ gives (47). It remains to estimate the total number of iterations K required by Moshpit SGD to find an ε-solution, i.e., to achieve $\mathbb{E}[f(\overline{\theta}^K) - f(\theta^*)] \leq \varepsilon$. Applying Lemma D.2 to (46), we get the following result: if $\mu > 0$ and

$$\gamma = \min\left\{\frac{1}{4L}, \frac{\ln\left(\max\left\{2, \min\left\{\frac{R_0^2\mu^2(1-\delta_{pv,1})^2K^2}{\delta_{pv,2}^2 + \sigma^2/N_{\min}}, \frac{R_0^2\mu^3(1-\delta_{pv,1})^3K^3}{3L\left(4\delta_{aq}^2 + (\tau-1)\sigma^2\right)}\right\}\right\}\right)}{(1-\delta_{pv,1})\mu K}\right\},$$

then $\mathbb{E}\left[f(\overline{\theta}^K) - f(\theta^*)\right]$ equals

$$\widetilde{\mathcal{O}}\left(LR_0^2\exp\left(-\frac{\mu}{L}(1-\delta_{pv,1})K\right) + \frac{\delta_{pv,2}^2 + \sigma^2/N_{\min}}{(1-\delta_{pv,1})\mu K} + \frac{L\left(\delta_{aq}^2 + (\tau-1)\sigma^2\right)}{(1-\delta_{pv,1})^2\mu^2K^2}\right),$$

implying (48). Similarly, we apply Lemma D.3 to (47) and get that for $\mu = 0$ and

$$\gamma = \min\left\{\frac{1}{4L}\sqrt{\frac{R_0}{(\delta_{pv,2}^2 + \sigma^2/N_{\min})K}}, \sqrt[3]{\frac{R_0^2}{3L\left(4\delta_{aq}^2 + (\tau-1)\sigma^2\right)K}}\right\},$$

$$\mathbb{E}\left[f(\overline{\theta}^K) - f(\theta^*)\right] = \mathcal{O}\left(\frac{LR_0^2}{K} + \sqrt{\frac{R_0^2(\delta_{pv,2}^2 + \sigma^2/N_{\min})}{K}} + \frac{\sqrt[3]{R_0^4 L\left(\delta_{aq}^2 + (\tau-1)\sigma^2\right)}}{K^{2/3}}\right),$$

implying (49). □

D.3 Non-Convex Case

In this section, we give the full proof of Theorem 3.4 about convergence of Moshpit SGD for general non-convex problems. The proof follows the similar steps as in the state-of-the-art analysis of Local-SGD in non-convex case [64, 63]. We start with the following lemma:

Lemma D.7. *Let $f_1 = \ldots = f_N = f$, function f be L-smooth and bounded from below by f_*, and Assumptions 3.1 and 3.2 hold with $\Delta_{pv}^k = \delta_{pv,1}\gamma\mathbb{E}[\|\nabla f(\theta^k)\|^2] + L\gamma^2\delta_{pv,2}^2$, $\delta_{pv,1} \in [0, 1/2)$, $\delta_{pv,2} \geq 0$. Then, for any $K \geq 0$ the iterates produced by Moshpit SGD with $\gamma \leq (1-2\delta_{pv,1})/8L$ satisfy*

$$\begin{aligned}\frac{(1-2\delta_{pv,1})\gamma}{4}\sum_{k=0}^{K-1}\mathbb{E}\left[\|\nabla f(\theta^k)\|^2\right] &\leq f(\theta^0) - f_* + \gamma L^2\sum_{k=0}^{K-1}\mathbb{E}[V_k] \\ &\quad + KL\gamma^2\left(\frac{\sigma^2}{N_{\min}} + \delta_{pv,2}^2\right),\end{aligned} \tag{50}$$

where $V_k = \frac{1}{N_k}\sum_{i\in P_k}\|\theta_i^k - \theta^k\|^2$ and $\theta^k = \frac{1}{N_k}\sum_{i\in P_k}\theta_i^k$.

Proof. Recall that Assumption 3.2 with $\Delta_{pv}^k = \delta_{pv,1}\gamma\mathbb{E}[\|\nabla f(\theta^k)\|^2] + L\gamma^2\delta_{pv,2}^2$ states

$$\mathbb{E}\left[\langle\nabla f(\theta^k), \theta^{k+1} - \widehat{\theta}^{k+1}\rangle + L\|\widehat{\theta}^{k+1} - \theta^{k+1}\|^2\right] \leq \delta_{pv,1}\gamma\mathbb{E}[\|\nabla f(\theta^k)\|^2] + L\gamma^2\delta_{pv,2}^2, \tag{51}$$

where $\widehat{\theta}^{k+1} = \frac{1}{N_k}\sum_{i\in P_k}(\theta_i^k - \gamma g_i^k)$. As for the convex case, the definition of $\widehat{\theta}^{k+1}$ implies

$$\widehat{\theta}^{k+1} = \frac{1}{N_k}\sum_{i\in P_k}\theta_i^k - \frac{\gamma}{N_k}\sum_{i\in P_k}g_i^k = \theta^k - \gamma g^k,$$

where $g^k = \frac{1}{N_k}\sum_{i\in P_k}g_i^k$. Using this and L-smoothness of f, we derive

$$\begin{aligned}f(\theta^{k+1}) - f(\theta^k) &\overset{(29)}{\leq} \langle\nabla f(\theta^k), \theta^{k+1} - \theta^k\rangle + \frac{L}{2}\|\theta^{k+1} - \theta^k\|^2 \\ &\overset{(32)}{\leq} \langle\nabla f(\theta^k), \widehat{\theta}^{k+1} - \theta^k\rangle + \langle\nabla f(\theta^k), \theta^{k+1} - \widehat{\theta}^{k+1}\rangle \\ &\qquad + L\|\widehat{\theta}^{k+1} - \theta^k\|^2 + L\|\theta^{k+1} - \widehat{\theta}^{k+1}\|^2 \\ &= -\gamma\langle\nabla f(\theta^k), g^k\rangle + L\gamma^2\|g^k\|^2 + \langle\nabla f(\theta^k), \theta^{k+1} - \widehat{\theta}^{k+1}\rangle \\ &\qquad + L\|\theta^{k+1} - \widehat{\theta}^{k+1}\|^2,\end{aligned}$$

from which it follows that

$$\begin{aligned}\mathbb{E}\left[f(\theta^{k+1}) - f(\theta^k) \mid \theta^k\right] &\leq -\gamma\left\langle\nabla f(\theta^k), \frac{1}{N_k}\sum_{i\in P_k}\nabla f(\theta_i^k)\right\rangle \\ &\quad + \mathbb{E}\left[\langle\nabla f(\theta^k), \theta^{k+1} - \widehat{\theta}^{k+1}\rangle \mid \theta^k\right] \\ &\quad + \mathbb{E}\left[L\|\theta^{k+1} - \widehat{\theta}^{k+1}\|^2 \mid \theta^k\right] \\ &\quad + L\gamma^2\mathbb{E}\left[\left\|\frac{1}{N_k}\sum_{i\in P_k}g_i^k\right\|^2 \mid \theta^k\right],\end{aligned} \tag{52}$$

where $\mathbb{E}\left[\,\cdot\mid\theta^k\right] := \mathbb{E}\left[\,\cdot\mid P_k, \theta_i^k, i\in P_k\right]$. Next, we estimate the last three terms in the right-hand side of (52). First of all,

$$\begin{aligned}
-\gamma\left\langle\nabla f(\theta^k), \frac{1}{N_k}\sum_{i\in P_k}\nabla f(\theta_i^k)\right\rangle &= -\gamma\|\nabla f(\theta^k)\|^2\\
&\quad -\gamma\left\langle\nabla f(\theta^k), \frac{1}{N_k}\sum_{i\in P_k}\nabla f(\theta_i^k)-\nabla f(\theta^k)\right\rangle\\
&\overset{(34)}{\leq} -\gamma\|\nabla f(\theta^k)\|^2+\frac{\gamma}{2}\|\nabla f(\theta^k)\|^2\\
&\quad +\frac{\gamma}{2}\left\|\frac{1}{N_k}\sum_{i\in P_k}(\nabla f(\theta_i^k)-\nabla f(\theta^k))\right\|^2\\
&\overset{(33)}{\leq} -\frac{\gamma}{2}\|\nabla f(\theta^k)\|^2+\frac{\gamma}{2N_k}\sum_{i\in P_k}\|\nabla f(\theta_i^k)-\nabla f(\theta^k)\|^2\\
&\overset{(28)}{\leq} -\frac{\gamma}{2}\|\nabla f(\theta^k)\|^2+\frac{\gamma L^2}{2}V_k, \qquad (53)
\end{aligned}$$

where $V_k = \frac{1}{N_k}\sum_{i\in P_k}\|\theta_i^k-\theta^k\|^2$. Secondly, since the stochastic gradients $\{g_i^k\}_{i\in P_k}$ are computed independently, we derive

$$\begin{aligned}
L\gamma^2\mathbb{E}\left[\left\|\frac{1}{N_k}\sum_{i\in P_k}g_i^k\right\|^2\mid\theta^k\right] &\overset{(35)}{=} L\gamma^2\left\|\frac{1}{N_k}\sum_{i\in P_k}\nabla f(\theta_i^k)\right\|^2\\
&\quad +L\gamma^2\mathbb{E}\left[\left\|\frac{1}{N_k}\sum_{i\in P_k}(g_i^k-\nabla f(\theta_i^k))\right\|^2\mid\theta^k\right]\\
&\overset{(33)}{\leq} 2L\gamma^2\left\|\frac{1}{N_k}\sum_{i\in P_k}(\nabla f(\theta_i^k)-\nabla f(\theta^k))\right\|^2\\
&\quad +2L\gamma^2\|\nabla f(\theta^k)\|^2\\
&\quad +\frac{\gamma^2 L}{N_k^2}\sum_{i\in P_k}\mathbb{E}\left[\|g_i^k-\nabla f(\theta_i^k)\|^2\mid\theta^k\right]\\
&\overset{(33),(7)}{\leq} \frac{2\gamma^2 L}{N_k}\sum_{i\in P_k}\|\nabla f(\theta_i^k)-\nabla f(\theta^k)\|^2\\
&\quad +2L\gamma^2\|\nabla f(\theta^k)\|^2+\frac{\gamma^2L\sigma^2}{N_k}\\
&\overset{(28)}{\leq} \underbrace{\frac{2L^3\gamma^2}{N_k}\sum_{i\in P_k}\|\theta_i^k-\theta^k\|^2}_{2L^3\gamma^2V_k}+2L\gamma^2\|\nabla f(\theta^k)\|^2\\
&\quad +\frac{\gamma^2L\sigma^2}{N_{\min}}. \qquad (54)
\end{aligned}$$

Plugging (53) and (54) in (52), we obtain

$$\begin{aligned}
\mathbb{E}\left[f(\theta^{k+1})-f(\theta^k)\mid\theta^k\right] &\leq -\frac{\gamma}{2}\left(1-4L\gamma\right)\|\nabla f(\theta^k)\|^2+\frac{\gamma L^2}{2}\left(1+4L\gamma\right)V_k+\frac{L\gamma^2\sigma^2}{N_{\min}}\\
&\quad +\mathbb{E}\left[\langle\nabla f(\theta^k), \theta^{k+1}-\widehat{\theta}^{k+1}\rangle+L\|\theta^{k+1}-\widehat{\theta}^{k+1}\|^2\mid\theta^k\right].
\end{aligned}$$

Next, we take the full expectation from the both sides of the above inequality, apply the tower property (36) and take into account that $\gamma \leq {}^{(1-2\delta_{pv,1})}/_{8L}$:

$$
\begin{aligned}
\mathbb{E}\left[f(\theta^{k+1}) - f(\theta^k)\right] &\leq -\frac{\gamma}{2}\left(1 - 4L\gamma\right)\mathbb{E}\left[\|\nabla f(\theta^k)\|^2\right] + \frac{\gamma L^2}{2}\left(1 + 4L\gamma\right)\mathbb{E}[V_k] + \frac{L\gamma^2\sigma^2}{N_{\min}} \\
&\quad +\mathbb{E}\left[\langle \nabla f(\theta^k), \theta^{k+1} - \widehat{\theta}^{k+1}\rangle + L\|\theta^{k+1} - \widehat{\theta}^{k+1}\|^2\right] \\
&\overset{(51)}{\leq} -\frac{\gamma}{2}\left(1 - 2\delta_{pv,1} - 4L\gamma\right)\mathbb{E}\left[\|\nabla f(\theta^k)\|^2\right] + \frac{\gamma L^2}{2}\left(1 + 4L\gamma\right)\mathbb{E}[V_k] \\
&\quad +L\gamma^2\left(\frac{\sigma^2}{N_{\min}} + \delta_{pv,2}^2\right) \\
&\leq -\frac{(1-2\delta_{pv,1})\gamma}{4}\mathbb{E}\left[\|\nabla f(\theta^k)\|^2\right] + \gamma L^2\mathbb{E}[V_k] \\
&\quad +L\gamma^2\left(\frac{\sigma^2}{N_{\min}} + \delta_{pv,2}^2\right).
\end{aligned}
$$

Summing up the obtained inequalities for $k = 0, \ldots, K-1$ and rearranging the terms, we derive

$$
\begin{aligned}
\frac{(1-2\delta_{pv,1})\gamma}{4}\sum_{k=0}^{K-1}\mathbb{E}\left[\|\nabla f(\theta^k)\|^2\right] &\leq \sum_{k=0}^{K-1}\mathbb{E}\left[f(\theta^k) - f(\theta^{k+1})\right] + \gamma L^2\sum_{k=0}^{K-1}\mathbb{E}[V_k] \\
&\quad +KL\gamma^2\left(\frac{\sigma^2}{N_{\min}} + \delta_{pv,2}^2\right) \\
&= f(\theta^0) - \mathbb{E}[f(\theta^K)] + \gamma L^2\sum_{k=0}^{K-1}\mathbb{E}[V_k] \\
&\quad +KL\gamma^2\left(\frac{\sigma^2}{N_{\min}} + \delta_{pv,2}^2\right) \\
&\leq f(\theta^0) - f_* + \gamma L^2\sum_{k=0}^{K-1}\mathbb{E}[V_k] \\
&\quad +KL\gamma^2\left(\frac{\sigma^2}{N_{\min}} + \delta_{pv,2}^2\right),
\end{aligned}
$$

where f_* is a uniform lower bound for f. □

The next step towards completing the proof of Theorem 3.4 gives the upper bound for $\sum_{k=0}^{K-1}\mathbb{E}[V_k]$ that appeared in (50).

Lemma D.8. *Let $f_1 = \ldots = f_N = f$ be L-smooth and bounded from below by f_*, and Assumptions 3.1 and 3.2 hold with $\Delta_{pv}^k = \delta_{pv,1}\gamma\mathbb{E}[\|\nabla f(\theta^k)\|^2] + L\gamma^2\delta_{pv,2}^2$, $\delta_{pv,1} \in [0, {}^1/_2)$, $\delta_{pv,2} \geq 0$. Then, for any $K \geq 0$ the iterates produced by Moshpit SGD with $\gamma \leq {}^1/_{\left(4\sqrt{e}L(\tau-1)\right)}$ satisfy*

$$
\sum_{k=0}^{K-1}\mathbb{E}[V_k] \leq 8e\gamma^2(\tau-1)^2\sum_{k=0}^{K-1}\mathbb{E}[\|\nabla f(\theta^k)\|^2] + 4\gamma^2K\left(2\delta_{aq}^2 + e(\tau-1)\sigma^2\right), \tag{55}
$$

where $V_k = \frac{1}{N_k}\sum_{i\in P_k}\|\theta_i^k - \theta^k\|^2$ and $\theta^k = \frac{1}{N_k}\sum_{i\in P_k}\theta_i^k$.

Proof. First of all, consider k such that $k = a\tau + t'$ for some $t' \in [0, \tau)$. Let $\mathbb{E}_g[\cdot]$ denote the expectation conditioned on $\{P_t\}_{t=a\tau}^{(a+1)\tau-1}$. Then

$$\begin{aligned}
\mathbb{E}_g[V_k] &= \frac{1}{N_k}\sum_{i\in P_k}\mathbb{E}_g\left[\|\theta_i^k - \theta^k\|^2\right] \overset{(35)}{\leq} \frac{1}{N_k}\sum_{i\in P_k}\mathbb{E}_g\left[\|\theta_i^k - \theta^{a\tau}\|^2\right] \\
&= \frac{1}{N_k}\sum_{i\in P_k}\mathbb{E}_g\left[\left\|\theta_i^{a\tau} - \theta^{a\tau} - \gamma\sum_{t=a\tau}^{k-1} g_i^t\right\|^2\right] \\
&\overset{(32)}{\leq} \frac{2}{N_k}\sum_{i\in P_k}\mathbb{E}_g\left[\|\theta_i^{a\tau} - \theta^{a\tau}\|^2\right] + \frac{2\gamma^2}{N_k}\sum_{i\in P_k}\mathbb{E}_g\left[\left\|\sum_{t=a\tau}^{k-1} g_i^t\right\|^2\right]. \qquad (56)
\end{aligned}$$

Next, we estimate the second term in the right-hand side of (56) using Lemma D.4:

$$\begin{aligned}
\frac{2\gamma^2}{N_k}\sum_{i\in P_k}\mathbb{E}_g\left[\left\|\sum_{t=a\tau}^{k-1} g_i^t\right\|^2\right] &\overset{(38)}{\leq} \frac{2e\gamma^2(k-a\tau)}{N_k}\sum_{i\in P_k}\sum_{t=a\tau}^{k-1}\mathbb{E}_g[\|\nabla f(\theta_i^t)\|^2] \\
&\quad + \frac{2e\gamma^2}{N_k}\sum_{i\in P_k}\sum_{t=a\tau}^{k-1}\mathbb{E}_g[\|g_i^t - \nabla f(\theta_i^t)\|^2] \\
&\overset{(32),(7)}{\leq} 4e\gamma^2(\tau-1)\sum_{t=a\tau}^{k-1}\mathbb{E}_g[\|\nabla f(\theta^t)\|^2] \\
&\quad + 4e\gamma^2(\tau-1)\sum_{t=a\tau}^{k-1}\frac{1}{N_k}\sum_{i\in P_k}\mathbb{E}_g[\|\nabla f(\theta_i^t) - \nabla f(\theta^t)\|^2] \\
&\quad + 2e\gamma^2(k-a\tau)\sigma^2 \\
&\overset{(28)}{\leq} 4e\gamma^2(\tau-1)\sum_{t=a\tau}^{k-1}\mathbb{E}_g[\|\nabla f(\theta^t)\|^2] \\
&\quad + 4e\gamma^2L^2(\tau-1)\sum_{t=a\tau}^{k-1}\frac{N_t}{N_k}\cdot\frac{1}{N_t}\sum_{i\in P_t}\mathbb{E}_g[\|\theta_i^t - \theta^t\|^2] \\
&\quad + 2e\gamma^2(\tau-1)\sigma^2 \\
&\leq 4e\gamma^2(\tau-1)\sum_{t=a\tau}^{k-1}\mathbb{E}_g[\|\nabla f(\theta^t)\|^2] \\
&\quad + 8e\gamma^2L^2(\tau-1)\sum_{t=a\tau}^{k-1}\mathbb{E}_g[V_t] + 2e\gamma^2(\tau-1)\sigma^2,
\end{aligned}$$

where in the last two inequalities we use $N_k = |P_k| \leq |P_{k-1}| = N_{k-1}$ for all $k \geq 1$ and $N_{a\tau} \leq 2N_{(a+1)\tau}$ for all integer $a \geq 0$. Plugging this inequality in (56) and taking the full expectation from the result, we get

$$\begin{aligned}
\mathbb{E}[V_k] &\le 2\mathbb{E}\left[\frac{1}{N_k}\sum_{i\in P_k}\|\theta_i^{a\tau}-\theta^{a\tau}\|^2\right] + 4e\gamma^2(\tau-1)\sum_{t=a\tau}^{k-1}\mathbb{E}[\|\nabla f(\theta^t)\|^2] \\
&\quad +8e\gamma^2L^2(\tau-1)\sum_{t=a\tau}^{k-1}\mathbb{E}[V_t] + 2e\gamma^2(\tau-1)\sigma^2 \\
&\le 4\mathbb{E}\left[\frac{1}{N_{a\tau}}\sum_{i\in P_{a\tau}}\|\theta_i^{a\tau}-\theta^{a\tau}\|^2\right] + 4e\gamma^2(\tau-1)\sum_{t=a\tau}^{k-1}\mathbb{E}[\|\nabla f(\theta^t)\|^2] \\
&\quad +8e\gamma^2L^2(\tau-1)\sum_{t=a\tau}^{k-1}\mathbb{E}[V_t] + 2e\gamma^2(\tau-1)\sigma^2 \\
&\overset{(10)}{\le} 4e\gamma^2(\tau-1)\sum_{t=a\tau}^{k-1}\mathbb{E}[\|\nabla f(\theta^t)\|^2] + 8e\gamma^2L^2(\tau-1)\sum_{t=a\tau}^{k-1}\mathbb{E}[V_t] \\
&\quad +2\gamma^2\left(2\delta_{aq}^2 + e(\tau-1)\sigma^2\right),
\end{aligned}$$

where in the second inequality we also use $N_k = |P_k| \le |P_{k-1}| = N_{k-1}$ for all $k \ge 1$ and $N_{a\tau} \le 2N_{(a+1)\tau}$ for all integer $a \ge 0$. Summing up the obtained inequalities for $k = a\tau, a\tau+1, \ldots, K'$ for some $K' \in [a\tau, (a+1)\tau - 1]$ we derive

$$\begin{aligned}
\sum_{k=a\tau}^{K'}\mathbb{E}[V_k] &\le 4e\gamma^2(\tau-1)\sum_{k=a\tau}^{K'}\sum_{t=a\tau}^{k-1}\mathbb{E}[\|\nabla f(\theta^t)\|^2] + 8e\gamma^2L^2(\tau-1)\sum_{k=a\tau}^{K'}\sum_{t=a\tau}^{k-1}\mathbb{E}[V_t] \\
&\quad +2\gamma^2(K'-a\tau+1)\left(2\delta_{aq}^2 + e(\tau-1)\sigma^2\right) \\
&\le 4e\gamma^2(\tau-1)^2\sum_{k=a\tau}^{K'}\mathbb{E}[\|\nabla f(\theta^k)\|^2] + 8e\gamma^2L^2(\tau-1)^2\sum_{k=a\tau}^{K'}\mathbb{E}[V_k] \\
&\quad +2\gamma^2(K'-a\tau+1)\left(2\delta_{aq}^2 + e(\tau-1)\sigma^2\right) \\
&\le 4e\gamma^2(\tau-1)^2\sum_{k=a\tau}^{K'}\mathbb{E}[\|\nabla f(\theta^k)\|^2] + \frac{1}{2}\sum_{k=a\tau}^{K'}\mathbb{E}[V_k] \\
&\quad +2\gamma^2(K'-a\tau+1)\left(2\delta_{aq}^2 + e(\tau-1)\sigma^2\right),
\end{aligned}$$

where in the last inequality we use $\gamma \le 1/(4\sqrt{e}L(\tau-1))$. Rearranging the terms, we get that for $K' \ge 0$

$$\sum_{k=a\tau}^{K'}\mathbb{E}[V_k] \le 8e\gamma^2(\tau-1)^2\sum_{k=a\tau}^{K'}\mathbb{E}[\|\nabla f(\theta^k)\|^2] + 4\gamma^2(K'-a\tau+1)\left(2\delta_{aq}^2 + e(\tau-1)\sigma^2\right),$$

where $a \ge 0$ is an integer such that $a\tau \le K' \le (a+1)\tau - 1$. Summing up the obtained inequalities for $K' = \tau-1, 2\tau-1, \ldots, \tau\lfloor (K-1)/\tau\rfloor - 1, K-1$, we derive (55). □

Combining Lemmas D.7 and D.8, we get the following result:

Theorem D.2 (Theorem 3.4). *Let $f_1 = \ldots = f_N = f$, function f be L-smooth and bounded from below by f_*, and Assumptions 3.1 and 3.2 hold with $\Delta_{pv}^k = \delta_{pv,1}\gamma\mathbb{E}[\|\nabla f(\theta^k)\|^2] + L\gamma^2\delta_{pv,2}^2$, $\delta_{pv,1} \in [0, 1/2)$, $\delta_{pv,2} \ge 0$. Then, for any $K \ge 0$ the iterates produced by Moshpit SGD with*

$$\gamma \le \min\left\{\frac{1-2\delta_{pv,1}}{8L}, \frac{\sqrt{1-2\delta_{pv,1}}}{8\sqrt{e}L(\tau-1)}\right\}$$

satisfy

$$\begin{aligned}
\mathbb{E}\left[\|\nabla f(\theta_{rand}^K)\|^2\right] &\le \frac{8\Delta_0}{(1-2\delta_{pv,1})K\gamma} \\
&\quad +\frac{8L\gamma}{1-2\delta_{pv,1}}\left(\frac{\sigma^2}{N_{\min}} + \delta_{pv,2}^2 + 4\gamma L\left(2\delta_{aq}^2 + e(\tau-1)\sigma^2\right)\right), \quad (57)
\end{aligned}$$

where $\Delta_0 = f(\theta^0) - f_*$ and θ^K_{rand} is chosen uniformly at random from $\{\theta^0, \theta^1, \ldots, \theta^{K-1}\}$. That is, Moshpit SGD achieves $\mathbb{E}\left[\|\nabla f(\theta^K_{rand})\|^2\right] \leq \varepsilon^2$ after

$$\mathcal{O}\Bigg(\frac{L\Delta_0}{(1-2\delta_{pv,1})^2\varepsilon^2}\Bigg[1 + (\tau-1)\sqrt{1-2\delta_{pv,1}} + \frac{\delta_{pv,2}^2 + {}^{\sigma^2}\!/_{N_{\min}}}{\varepsilon^2} + \frac{\sqrt{(1-2\delta_{pv,1})(\delta_{aq}^2+(\tau-1)\sigma^2)}}{\varepsilon}\Bigg]\Bigg) \tag{58}$$

iterations with

$$\gamma = \min\left\{\frac{1-2\delta_{pv,1}}{8L}, \frac{\sqrt{1-2\delta_{pv,1}}}{8\sqrt{e}L(\tau-1)}, \sqrt{\frac{\Delta_0}{LK\left(\delta_{pv,2}^2 + {}^{\sigma^2}\!/_{N_{\min}}\right)}}, \sqrt[3]{\frac{\Delta_0}{4L^2\left(2\delta_{aq}^2 + e(\tau-1)\sigma^2\right)}}\right\}.$$

Proof of Theorem 3.4. Plugging the result of Lemma D.8 in the inequality (50) from Lemma D.7, we obtain

$$\begin{aligned}\frac{(1-2\delta_{pv,1})\gamma}{4}\sum_{k=0}^{K-1}\mathbb{E}\left[\|\nabla f(\theta^k)\|^2\right] &\leq f(\theta^0) - f_* + 8e\gamma^3L^2\tau(\tau-1)\sum_{k=0}^{K-1}\mathbb{E}[\|\nabla f(\theta^k)\|^2] \\ &\quad + KL\gamma^2\left(\frac{\sigma^2}{N_{\min}} + \delta_{pv,2}^2\right) \\ &\quad + 4KL^2\gamma^3\left(2\delta_{aq}^2 + e(\tau-1)\sigma^2\right) \\ &\leq f(\theta^0) - f_* + \frac{(1-2\delta_{pv,1})\gamma}{8}\sum_{k=0}^{K-1}\mathbb{E}\left[\|\nabla f(\theta^k)\|^2\right] \\ &\quad + KL\gamma^2\left(\frac{\sigma^2}{N_{\min}} + \delta_{pv,2}^2\right) \\ &\quad + 4KL^2\gamma^3\left(2\delta_{aq}^2 + e(\tau-1)\sigma^2\right).\end{aligned}$$

Next,

$$\begin{aligned}\frac{1}{K}\sum_{k=0}^{K}\mathbb{E}\left[\|\nabla f(\theta^k)\|^2\right] &\leq \frac{8\Delta_0}{(1-2\delta_{pv,1})K\gamma} \\ &\quad + \frac{8L\gamma}{1-2\delta_{pv,1}}\left(\frac{\sigma^2}{N_{\min}} + \delta_{pv,2}^2 + 4\gamma L\left(2\delta_{aq}^2 + e(\tau-1)\sigma^2\right)\right),\end{aligned}$$

where $\Delta_0 = f(\theta^0) - f_*$. Since θ^K_{rand} is chosen uniformly at random from $\{\theta^0, \theta^1, \ldots, \theta^{K-1}\}$,

$$\mathbb{E}\left[\|\nabla f(\theta^K_{\text{rand}})\|^2\right] \overset{(36)}{=} \frac{1}{K}\sum_{k=0}^{K}\mathbb{E}\left[\|\nabla f(\theta^k)\|^2\right]$$

and (57) holds. Applying Lemma D.3 to (57), we get the following result: if

$$\gamma = \min\left\{\frac{1-2\delta_{pv,1}}{8L}, \frac{\sqrt{1-2\delta_{pv,1}}}{8\sqrt{e}L(\tau-1)}, \sqrt{\frac{\Delta_0}{LK\left(\delta_{pv,2}^2 + {}^{\sigma^2}\!/_{N_{\min}}\right)}}, \sqrt[3]{\frac{\Delta_0}{4L^2\left(2\delta_{aq}^2 + e(\tau-1)\sigma^2\right)}}\right\},$$

then $\mathbb{E}\left[\|\nabla f(\theta^K_{\text{rand}})\|^2\right]$ equals

$$\mathcal{O}\left(\frac{L\Delta_0\left(1+(\tau-1)\sqrt{1-2\delta_{pv,1}}\right)}{(1-2\delta_{pv,1})^2K} + \sqrt{\frac{L\Delta_0\left(\delta_{pv,2}^2 + {}^{\sigma^2}\!/_{N_{\min}}\right)}{(1-2\delta_{pv,1})^2K}} + \frac{\sqrt[3]{L^2\Delta_0^2(\delta_{aq}^2+(\tau-1)\sigma^2)}}{(1-2\delta_{pv,1})K^{2/3}}\right),$$

which implies the desired convergence result from (58). □

E Decentralized matchmaking

In order to run group all-reduce over unreliable devices, Moshpit Averaging must be able to dynamically form groups of active devices that share the same key C_i. In theory, this matchmaking can be implemented precisely as described in Algorithm 1: each peer adds itself to a certain DHT key, waits for a said period of time, and then reads the same key to retrieve a list of its groupmates.

However, in practice, this kind of matchmaking would be extremely fragile: if any peer arrives late (for example, due to latency), it may join the group when other peers have already finished matchmaking. As a result, some workers will treat this peer as active, while others will behave as though there is no such peer at all, breaking the consensus and rendering all peers unable to run all-reduce in a stable manner.

To avoid this and other similar inconsistencies, Moshpit All-Reduce employs a more sophisticated matchmaking protocol with the following guarantees

1. Peers that join the same group are guaranteed to have the same list of groupmates;
2. The group will have the maximum possible number of peers, unless some of them fail;
3. If some peers fail, matchmaking will still form the group out of the remaining ones.

To achieve this, each peer first declares itself onto the DHT (as in Algorithm 1). Then, peers attempt to form groups by calling the `REQUEST_JOIN_GROUP` remote procedure call. Intuitively, if peer A calls this RPC on peer B, then *peer A requests to join peer B's group*, which can be either accepted or rejected by the group "leader" B, which may or may not have other "followers".

If a peer is accepted to a group, it commits to stay active (i.e. to await other peers) for a set period of time and perform all-reduce with the peers supplied by the group "leader". On the other hand, a peer can be rejected if (a) the potential "leader" is already a follower in another group, (b) the group is already running all-reduce, or (c) if the "leader" failed or left during matchmaking.

To ensure that this protocol forms groups of maximum size, each peer generates a unique "priority" based on its local timestamp[9]. Peers prioritize joining the group of neighbors that have the lowest "priority". Under normal circumstances, all workers will join the group of a peer that was first to start matchmaking according to its own local time. However, if this peer has failed or already finished matchmaking, the group will be formed around one of the remaining peers.

Matchmaking for 64 peers can take less than 1 second if all workers are located in the same cloud region and are highly synchronized. However, this can grow to 2.9 seconds for two different cloud regions and up to 9 seconds when training with commodity hardware around the world.

To ensure that this latency does not affect the training performance, Moshpit SGD performs matchmaking asynchronously in the background thread, while the model is accumulating gradients. All peers begin matchmaking 15 seconds before the estimated averaging round, so that in $\geq 95\%$ of averaging iterations, the matchmaking step is already finished by the time peers need to run all-reduce.

F Training with a dynamic number of peers

Many practical setups with unreliable devices allow peers to join or leave at any time, which can produce undesirable side-effects. For instance, consider a participant that joins the "swarm" midway through the training process. If this participant starts with the initial model parameters, it can undo some of the progress made by other peers.

To circumvent this issue, we require each new participant to download the latest parameters from a random up-to-date peer discovered through DHT. The same technique is used to synchronize the optimizer statistics and the learning rate schedule. This protocol is also triggered if a peer becomes desynchronized with others, e.g., after a network freeze.

[9] More specifically, the priority is a tuple of `(timestamp, peer_id)`, where `peer_id` is used to break ties.

G Load balancing via linear programming

When running Moshpit Averaging on heterogeneous devices, one must regularly perform Butterfly All-Reduce among peers with uneven network bandwidth. In order to speed up the protocol, we can make low-throughput peers receive, average, and send smaller partitions of the averaged vector; conversely, the high-throughput peers can process greater fractions of the input vector. To compute the optimal partitioning, peers must solve an optimization problem that minimizes the total time spent on communication during all-reduce.

Consider a group of M peers with network bandwidths $b_1, ..., b_M$, defined for simplicity as the minimum of the upload and download speed for each peer. Our objective is to find w_i — a fraction of all input vectors to be processed by the i-th peer.

In Butterfly All-Reduce, each peer i splits its vector into parts and sends these parts to corresponding peers. Since there is no need to send w_i to itself, i-th peer will upload a total of $1 - w_i$ of the vector to its peers. On the receiving side, peer i will average w_i of the vector from all peers in its group. To do so, it must download $M - 1$ vector parts of size w_i from all other peers. After that, peers distribute the averaged parts by running the same procedure in reverse (see Figure 1).

Thus, the communication time for each peer is proportional to $t_i = (1 - w_i + (M - 1)w_i) \cdot \frac{1}{b_i}$ and the total runtime of Butterfly All-Reduce is the maximum communication time over all peers: $T = \max_i t_i = \max_i (1 - w_i + (M - 1)w_i) \cdot \frac{1}{b_i}$. Formally, we minimize T with respect to w_i with two constraints on the fraction weights:

$$\begin{aligned} \min_{w} \quad & \max_{i}(1 - w_i + (M-1)w_i) \cdot \frac{1}{b_i} \\ \text{subject to} \quad & \sum_{i=1}^{M} w_i = 1 \\ & w_i \geq 0 \qquad \forall i = 1, \ldots, M \end{aligned}$$

Because the functions being maximized and the constraints are linear in w_i, this problem can be reduced to linear programming [125]. Namely, we can minimize a surrogate variable ξ such that $\forall i,\ \xi \geq (1 - w_i + (M - 1) \cdot w_i) \cdot \frac{1}{b_i}$. The resulting linear program is formulated as follows:

$$\begin{aligned} \min_{w,\xi} \quad & \xi \\ \text{subject to} \quad & \sum_{i=1}^{M} w_i = 1 \\ & w_i \geq 0 \qquad \forall i = 1, \ldots, M \\ & \xi \geq (1 - w_i + (M-1)w_i) \cdot \frac{1}{b_i} \quad \forall i = 1, \ldots, M \end{aligned}$$

We solve this problem using the interior point method [126] implemented as part of the SciPy package (`scipy.optimize.linprog`). Note that depending on the conditions given by participant bandwidth, optimal weights of specific peers might be equal to 0 in some cases. In essence, this allows our method to smoothly interpolate between data parallelism [9], parameter server [18] and sharded parameter server [25] in manner similar to BytePS [26].

H Detailed experimental setup

In this section, we provide the detailed hardware configuration of servers used for each of our distributed training experiments.

H.1 ImageNet training

Both homogeneous and heterogeneous training setups for ImageNet are provisioned in our on-premise infrastructure across multiple data centers and an office space (for the heterogeneous setup only).

Homogeneous. For the homogeneous setup, we use 16 identical instances with the following specifications:

- **GPU:** V100-PCIe,
- **CPU:** 6 vCPUs (Xeon E5-2650v4),
- **RAM:** 64GB.

Heterogeneous. In turn, the heterogeneous setup contains multiple instance types listed in Table 2:

Table 2: **Heterogeneous** setup for ImageNet training.

Instances	GPUs	GPU type	Cores	RAM, GB	CPU type
4	1	V100-PCIe	6	64	E5-2650v4
17	2	GTX 1080Ti	8	64	E5-2650v4
7	1	GTX 1080Ti	4	32	E5-2650v4
16	1	P40	4	32	E5-2667v2
20	1	M40-24GB	4	32	E5-2667v2

H.2 ALBERT training

Homogeneous. For the homogeneous setup, we use a single virtual machine with the following specifications:

- **GPU:** 8× V100-PCIe,
- **CPU:** 48 vCPUs (Xeon E5-2650v4),
- **RAM:** 488GB.

At the time of writing, the cloud rent cost for this instance is **$24.48** per hour.

Heterogeneous. Our heterogeneous setup is composed of two parts: AWS EC2 Spot instances and crowdsourced machines from the `Vast.ai` marketplace. For spot instances, we picked the smallest suitable instance size available from the cloud provider and further limited their bandwidth to 1Gb/s[10]. As for marketplace instances, we report the hardware specifications for each worker gathered 1 hour after the start of ALBERT training.

Since both cloud and marketplace instances are preemptible, the actual cost of the server fleet will vary based on the current price. For simplicity, we report the maximum hourly price we ended up paying for this instance (enforced via maximum bid). Finally, some marketplace instances have missing specifications, such as unknown CPU type. This is likely caused by non-standard virtualization configured by the device owner. The resulting fleet configuration, shown in Table 3, costs up to $15.43/hour, depending on the number of active instances.

I Additional averaging experiments

In this section, we evaluate the averaging precision with the same methodology as in 4.1, but for multiple different worker configurations.

Table 4 provides the complete results of our experiments that were used to make conclusions in the main experimental section: instead of reporting the mean squared error for different iterations, we provide the number of rounds that was required to achieve the error of 10^{-9} and 10^{-4}.

In Figure 5, plots 1–5 explore several combinations of grid sizes and failure rates, whereas plot 6 (bottom right) demonstrates a setup with the same number of peers (10^6) arranged into several different grid sizes and its relation to convergence. Note that M=32 outperforms the alternatives only for the specific failure rate of 0.001.

[10] We use `tc qdisc` Linux utility to artificially limit the network throughput, similarly to [127]

Table 3: **Heterogeneous** setup for ALBERT training.

GPU	Cores	RAM, GB	CPU type	Download, Mb/s	Upload, Mb/s	Cost, $/hour
			Preemptible `g4dn.xlarge` instances (32×)			
T4	4	16	Xeon Platinum 8259CL	1000	1000	0.1578
			Marketplace instances			
GTX 1070Ti	6	16	E5-2640	425	255	0.036
GTX 1070Ti	6	16	i3-6100T	121	36	0.06
GTX 1080Ti	4	20	i3-6096P	817	308	0.101
GTX 1080Ti	20	129	E5-2630v4	660	475	0.182
GTX 1080Ti	1	16	i7-7700K	245	210	0.302
GTX 1080Ti	48	97	Xeon Platinum 8124	583	539	0.217
GTX 1080Ti	10	16	Unknown	n/a	n/a	0.15
GTX 1080Ti	4	16	Xeon Gold 6149	98	100	0.2
GTX 1080Ti	4	16	Xeon Gold 6149	99	98	0.2
GTX 1080Ti	4	16	Xeon Gold 6149	99	99	0.2
GTX 1080Ti	4	16	Xeon Gold 6149	99	99	0.2
RTX 2070S	24	32	E5-2620v2	199	25	0.199
RTX 2070S	32	97	E5-2650	162	64	0.285
RTX 2080	6	16	E5-2620v3	271	287	0.25
RTX 2080	24	32	E5-2630v3	199	25	0.302
RTX 2080S	4	32	E5-2697v4	101	99	0.292
RTX 2080S	4	32	E5-2697v4	93	99	0.292
RTX 2080S	4	32	E5-2697v4	94	98	0.292
RTX 2080S	4	32	E5-2697v4	94	98	0.292
RTX 2080S	4	32	E5-2697v4	100	99	0.292
RTX 2080Ti	4	16	Ryzen Threadripper 3960x	279	271	0.35
RTX 2080Ti	8	129	E5-2670v3	616	672	0.201
RTX 2080Ti	6	32	E5-2620v3	217	61	0.22
RTX 2080Ti	8	16	E5-2697v2	100	58	0.3
RTX 2080Ti	8	21	E5-2697v2	145	49	0.243
RTX 2080Ti	12	32	Unknown	111	92	0.326
RTX 2080Ti	12	64	E5-2690v3	205	61	0.549
RTX 3080	16	16	i7-10700K	69	49	0.462
RTX 3090	14	32	E5-2695v3	93	37	0.498
RTX 3090	16	32	Ryzen 9 3950X	338	38	0.511
Titan RTX	4	32	Xeon W-3223	321	115	1
Titan RTX	4	32	Xeon Gold 6149	99	100	0.702
Titan V	8	32	i7-7700K	97	50	0.282
V100-FHHL	8	60	Xeon Gold 6148	544	584	0.39
			Total hourly cost (as listed):			**15.43**

J Additional image classification experiments

Aside from the two evaluation scenarios provided in 4.2, we also measure the performance of Moshpit-SGD in a non-distributed setup, i.e. on a single server with multiple GPUs. We conduct this experiment on the same 8× V100 machine that was used in the **homogeneous** setup for training ALBERT (see Appendix H.2).

As Figure 6 demonstrates, Moshpit SGD is slower than AR-SGD by approximately 25%. This result is expected, since our implementation of Moshpit All-Reduce is more general and communicates over a TCP connection, whereas AR-SGD uses direct peer-to-peer GPU communication over PCIe. On average, this incurs a slowdown of 27% in terms of training time.

Table 4: Averaging performance of different algorithms. Values denote the number of iterations required to achieve the error of 10^{-9} (10^{-4} in parentheses), the best result is in bold.

N	p	All-Reduce	Gossip	PushSum	Random groups	Moshpit
512	0	**1.0 (1.0)**	50.0 (50.0)	47.6 (15.6)	6.1 (3.0)	8.2 (3.5)
512	0.001	**1.6 (1.6)**	50.0 (50.0)	47.6 (15.6)	6.3 (3.0)	8.1 (3.7)
512	0.005	10.9 (10.9)	50.0 (50.0)	47.8 (15.6)	**6.3 (3.0)**	8.7 (3.9)
512	0.01	41.7 (41.7)	50.0 (50.0)	47.8 (15.6)	**6.6 (3.0)**	9.1 (3.9)
768	0	**1.0 (1.0)**	50.0 (50.0)	43.2 (13.8)	6.2 (3.0)	6.0 (3.0)
768	0.001	**1.8 (1.8)**	50.0 (50.0)	43.2 (13.8)	6.5 (3.0)	6.2 (3.0)
768	0.005	28.7 (28.7)	50.0 (50.0)	43.2 (14.1)	**6.6 (3.0)**	**6.6 (3.0)**
768	0.01	50.0 (50.0)	50.0 (50.0)	43.9 (14.2)	7.0 (3.0)	**6.8 (3.0)**
900	0	**1.0 (1.0)**	50.0 (50.0)	45.0 (14.7)	6.4 (3.0)	5.0 (2.8)
900	0.001	**1.8 (1.8)**	50.0 (50.0)	45.0 (14.7)	6.3 (3.0)	5.5 (3.0)
900	0.005	50.0 (50.0)	50.0 (50.0)	45.2 (14.7)	6.7 (3.0)	**5.9 (3.0)**
900	0.01	50.0 (50.0)	50.0 (50.0)	45.6 (14.9)	7.0 (3.1)	**6.4 (3.1)**
1024	0	**1.0 (1.0)**	50.0 (50.0)	49.0 (16.2)	6.2 (3.0)	2.0 (2.0)
1024	0.001	**2.0 (2.0)**	50.0 (50.0)	49.0 (16.3)	6.5 (3.0)	3.4 (2.2)
1024	0.005	42.6 (42.6)	50.0 (50.0)	49.5 (16.3)	6.7 (3.0)	**5.4 (2.9)**
1024	0.01	50.0 (50.0)	50.0 (50.0)	49.5 (16.3)	6.9 (3.1)	**5.9 (3.0)**

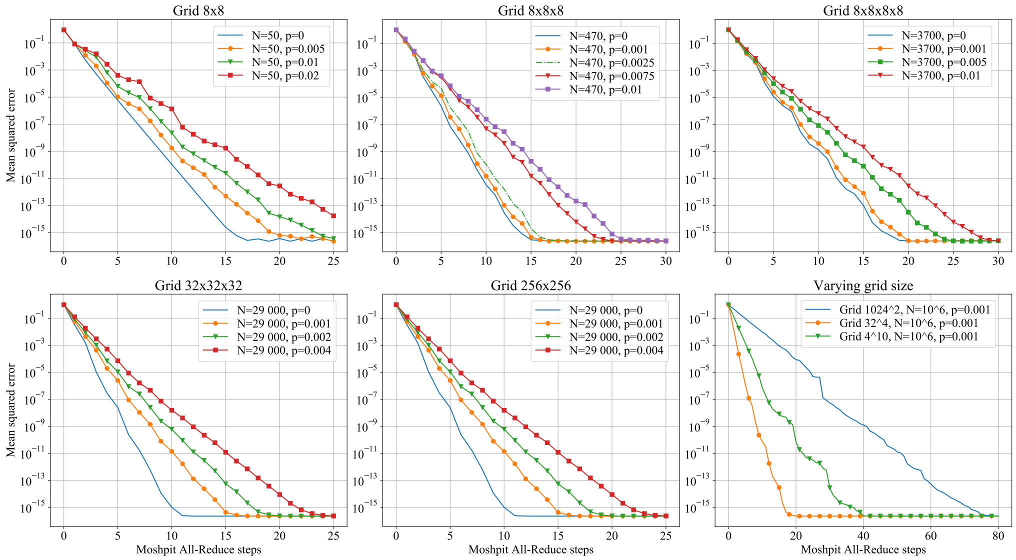

Figure 5: Averaging error of Moshpit All-Reduce as a function of the iteration number for different configurations and failure rates.

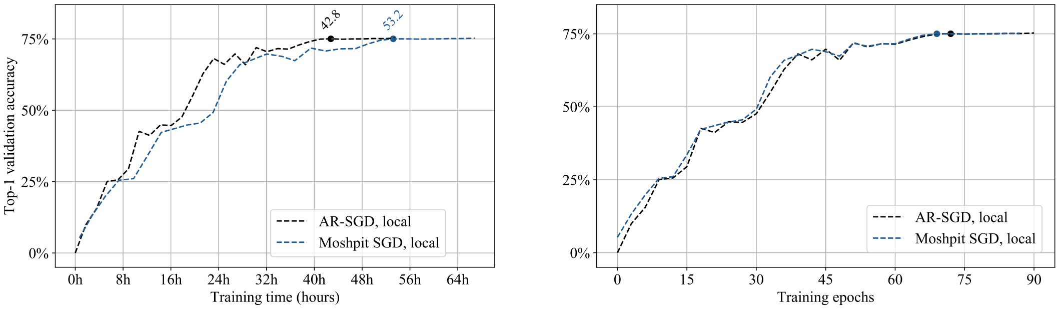

Figure 6: ResNet-50 top-1 validation accuracy on ImageNet when training on a single node with 8× V100-PCIe GPUs. **(Left)** Convergence in terms of training time, **(Right)** Convergence in terms of training epochs