SwarmSGD: Scalable Decentralized SGD with Local Updates

Giorgi Nadiradze
IST Austria

Amirmojtaba Sabour
IST Austria

Dan Alistarh
IST Austria

Aditya Sharma
IST Austria

Ilia Markov
IST Austria

Vitaly Aksenov
IST Austria

Abstract

The ability to scale distributed optimization to large node counts has been one of the main enablers of recent progress in machine learning. To this end, several techniques have been explored, such as asynchronous and decentralized execution–which significantly reduce the impact of communication and synchronization, and the ability for nodes to perform several local model updates before communicating–which reduces the frequency of communication.

In this paper, we show that these techniques, which have so far been considered independently, can be jointly leveraged to obtain near-perfect scalability for training neural network models via stochastic gradient descent (SGD). We consider a setting with minimal coordination: we have a large number of nodes on a communication graph, each with a local subset of data, performing independent SGD updates onto their local models. After some number of local updates, each node chooses an interaction partner uniformly at random from its neighbors, and averages its local model with the neighbor's model.

Our first contribution is in proving that, even under such a relaxed setting, SGD can still be guaranteed to converge to local minima under standard assumptions. The proof improves existing techniques by jointly handling decentralization, asynchrony, and local updates, and by bounding their impact. On the practical side, we instantiate this algorithm onto a supercomputing environment, and show that it can successfully converge and scale for large-scale image classification models, matching or even slightly improving the accuracy of the baseline parallel variants.

1 Introduction

Distributed machine learning has become commonplace, and it is not unusual to encounter systems which distribute model training among tens or even hundreds of nodes. In this paper, we take this trend to the extreme, and ask: would it be possible to distribute basic optimization procedures such as stochastic gradient

descent (SGD) to *thousands* of agents? How could the dynamics be implemented in such a large-scale setting, and what would be with the resulting convergence and speedup behavior?

For some intuition, let us consider the classical *data-parallel* distribution strategy for SGD [8]. We are in the classical empirical risk minimization setting, where we have a set of samples S from a distribution, and wish to minimize the function $f : \mathbb{R}^d \to \mathbb{R}$, which is the average of losses over samples from S by finding $x^\star = \operatorname{argmin}_x \sum_{s \in S} f_s(x)/|S|$. Assume that we have P compute nodes which can process samples in parallel. Data-parallel SGD consists of parallel iterations, in which each node computes the gradient for one sample, followed by a gradient exchange. Globally, this leads to the iteration:

$$x_{t+1} = x_t - \eta_t \sum_{i=1}^{P} \tilde{g}_t^i(x_t),$$

where η_t is the learning rate, x_t is the value of the global parameter, initially 0^d, and $\tilde{g}_t^i(x_t)$ is the stochastic gradient with respect to the parameter obtained by node i at time t.

When extending this strategy to high node counts, two major bottlenecks are *communication* and *synchronization*. In particular, to maintain a consistent view of the parameter x_t, the nodes would need to broadcast and receive all gradients, and would need to synchronize with all other nodes, at the end of every iteration. Significant work has been dedicated to removing these two barriers. In particular, there has been progress on *communication-reduced* variants of SGD (e.g. [31, 34, 4, 37, 3, 11, 15]), *asynchronous* variants (e.g. [28, 30, 12]), as well as *large-batch* or *periodic model averaging* methods, which aim to reduce the *frequency* of communication(e.g. [14, 39] and [10, 33]), or even *decentralized synchronous* variants (e.g. [23, 35, 20]). Using such techniques, it is possible to scale SGD to tens or even hundreds of nodes, even for complex objectives such as the training of deep neural networks. However, in systems with an order of magnitude more nodes, the communication and synchronization requirements of these algorithms may become infeasible.

In this paper, we investigate removing these scalability barriers by considering a drastically decoupled setting for parallel SGD: we are given a population of n compute agents, located at vertices of a connected graph, each of which can execute sequential SGD steps on its own local model, based on a fraction of the data. Periodically, after performing some number of local optimization steps, a node can initiate a pairwise interaction with a neighbor that is chosen *uniformly at random*. While decentralized SGD variants [23, 35, 6, 20], and synchronous local SGD variants with global periodic averaging [10, 33, 25] have been considered in previous work, we are the first to consider these decentralization and local updates in conjunction. The resulting algorithm allows the asynchrony afforded by the pairwise interactions to be combined with the reduced communication frequency given by multiple local SGD steps, leading to negligible overhead in terms of communication and synchronization. The key question is whether this extremely decentralized SGD variant could still possibly converge.

More precisely, in our algorithm each node i is assigned a set of samples S^i, and maintains its own parameter estimate x^i. Each node i performs local SGD steps on its model x^i based on its local data, and then picks a neighbor uniformly at random to share information with, via *averaging* of the two models. Effectively, if node i interacts with node j, node i's updated model becomes

$$x^i_{T+1} \leftarrow \frac{x^i_{T,H_i} + x^j_{T,H_j}}{2}, \tag{1}$$

where j is the interaction partner, and the input models $x^i_{T,H}$ and $x^j_{T,H}$ have been obtained by iterating the SGD H_i and H_j times, respectively, locally from the previous interaction of either node. The update for node j is symmetric, so that the two parameters match after the averaging step. In this paper, we analyze the above protocol, which we call SwarmSGD.

We show that, perhaps surprisingly, this simple decentralized SGD averaging dynamic with local updates provides strong convergence guarantees for non-convex objectives, under standard assumptions. Specifically, we show a $\Theta(\sqrt{n})$ speedup in the non-convex case, matching results from previous work which considered decentralized dynamics but which synchronize upon every SGD step, e.g. [23, 24]. Our analysis also extends to regular graph topologies, and generally shows that the impact of decentralization, asynchrony, and local updates can be asymptotically negligible in some parameter regimes.

On the practical side, we show that this algorithm can be easily mapped to a super-computing setting, where agents correspond to compute nodes, connected by a dense interconnection topology. Specifically, we apply SwarmSGD to train deep residual models [16] for CIFAR/ImageNet classification tasks [29, 21] deployed on the Piz Daint supercomputer [1]. Experiments confirm the perfect scalability of SwarmSGD. In addition, we observe an improvement in the *convergence* versus number of SGD iterations per model at higher node counts. Specifically, using SwarmSGD deployed on 32 GPU nodes, we are able to train the ResNet18 and ResNet50 [16] models to full accuracy on ImageNet using approximately 10 passes over the entire dataset ("epochs") per model, which is a reduction of 9× versus the sequential baseline. Even though collectively the nodes perform more dataset iterations than a classic sequential or data-parallel algorithm, our technique can result in significant end-to-end training time reduction compared to the data-parallel baseline, with perfect scalability.

Related Work. The study of decentralized optimization algorithms dates back to [36], and is related to the study of *gossip* algorithms for information dissemination [19, 38, 9]. Gossip is usually studied in one of two models [9]: *synchronous*, structured in global rounds, where each node interacts with a randomly chosen neighbor, and *asynchronous*, where each node wakes up at times given by a local Poisson clock, and picks a random neighbor to interact with. The model we consider can be seen as equivalent to the asynchronous gossip model. The key differences between our work and averaging in the gossip model, e.g. [9], are that that 1) we consider local SGD steps, which would not make sense in the case of averaging fixed initial values; and 2) the gossip input

model is *static* (node inputs are fixed, and node estimates must converge to the true mean), whereas we study a *dynamic* setting, where models are continually updated via SGD. Several optimization algorithms have been analyzed in this setting [27, 18, 32]. [35, 20] analyze quantization in the synchronous gossip model.

[23, 24, 6] consider SGD-type algorithms in gossip-like models. Specifically, they analyze the SGD averaging dynamic in the non-convex setting *but do not allow nodes to perform local updates*. In particular, nodes perform joint averaging upon every SGD step. Table 3 in the Appendix summarizes their assumptions, results, and rates. Their results are phrased in the synchronous gossip model, in which nodes interact in a sequence of perfect matchings, for which they provide $O(1/\sqrt{Tn})$ convergence rates under analytical assumptions. [24] extends these results to a variant of the gossip model where updates can be performed based on stale information.

Upon careful examination, one can find that their results can be extended to the asynchronous gossip setting we consider, *as long as nodes are not allowed to perform local SGD updates to their models (corresponding to $H = 1$)*. Extending the analysis of distributed SGD to allow for local steps is challenging even in *centralized* models, see for instance [33]. If we assume $H = 1$, our technique yields similar or better bounds relative to previous work in the decentralized model, as our potential analysis is specifically-tailored to this dynamic interaction model. For instance, for [6], the speedup with respect to the number of nodes depends on a parameter C, which in turn, depends on 1) the dimension of the objective function domain, 2) the number iterations for the graph given by edge sets of all matrices used in averaging to be connected, and the 3) diameter of the aforementioned connected graph. In the dynamic interaction model we consider, the parameter C will be at least *linear* in the number of nodes, which eliminates any speedup from this upper bound.

In sum, relative to prior work on decentralized algorithms, our contributions are as follows. We are the first to consider the impact of local updates in conjunction with decentralized SGD. We show that the cost for the linear reduction in communication in H given by this technique is at worst a squared convergence decrease in the parameter H. Our analysis technique relies on a fine-grained analysis of individual interactions, which is different than that of previous work, and can yield improved bounds even in the case where $H = 1$. From the implementation perspective, the performance of our algorithm is superior to that of previous methods, notably D-PSGD [23], AD-PSGD [24] and SGP [6]. We present a detailed comparison in the experimental section.

Other references which consider dynamic interaction models are Nedic et al. [26], who present a gradient tracking algorithm in a different dynamic graph model, and Hendrickx et al. [17], who achieve exponential convergence rates in a gossip model where transmissions are synchronized across *edges*. The algorithm they consider is a more complex instance of accelerated coordinate descent, and is therefore quite different from the simple dynamics we consider. Neither reference considers local updates, nor real-world scalability.

2 Preliminaries

The Distributed System Model. We consider a model which consists of a set of $n \geq 2$ anonymous agents, or nodes, each of which is able to perform local computation. We assume that communication network of nodes is a r-regular graph G with a spectral gap λ_2(second smallest eigenvalue of the Laplacian of G). The execution proceeds in discrete *steps*, where in each step we sample an edge of the graph G uniformly at random and we allow the agents corresponding to the edge endpoints interact. Each of the two chosen agents updates its state according to a state update function, specified by the algorithm. The basic unit of time is a single pairwise interaction between two nodes. Notice however that in a real system $\Theta(n)$ of these interactions could occur in parallel. Thus, a standard global measure is *parallel time*, defined as the total number of interactions divided by n, the number of nodes. Parallel time intuitively corresponds to the *average* number of interactions per node to convergence. We note that our model is virtually identical to the population model of distributed computing [5], or to asynchronous gossip models [38].

Stochastic Optimization. We assume that the agents wish to minimize a d-dimensional, differentiable function $f : \mathbb{R}^d \to \mathbb{R}$. Specifically, we will assume the empirical risk minimization setting, in which agents are given access to a set of data samples $S = \{s_1, \ldots, s_m\}$ coming from some underlying distribution $\mathcal{D}$, to a function $\ell_i : \mathbb{R}^d \to \mathbb{R}$ which encodes the loss of the argument at the sample s_i. The goal of the agents is to converge on a model x^* which minimizes the empirical loss, that is

$$x^* = \operatorname{argmin}_x f(x) = \operatorname{argmin}_x (1/m) \sum_{i=1}^{m} \ell_i(x). \tag{2}$$

In this paper, we assume that the agents employ these samples to run a decentralized variant of SGD, described in detail in the next section. For this, we will assume that each agent i has access to *stochastic gradients* $\widetilde{g}_i$ of the function f, which are functions such that $\mathbb{E}[\widetilde{g}_i(x)] = \nabla f(x)$. Stochastic gradients can be computed by each agent by sampling i.i.d. the distribution $\mathcal{D}$, and computing the gradient of f at θ with respect to that sample. (Our analysis can be extended to the case where each agent is sampling from its own partition of data, see Section F in the Appendix.) We will assume a subset of the following conditions about the objective function:

(1) **Smooth Gradients**: The gradient $\nabla f(x)$ is L-Lipschitz continuous for some $L > 0$, i.e. for all $x, y \in \mathbb{R}^d$:

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|. \tag{3}$$

(2) **Bounded Variance**: The variance of the stochastic gradients is bounded i.e. for all $x \in \mathbb{R}^d$ and agent i:

$$\mathbb{E}\left\|\widetilde{g}_i(x) - \nabla f(x)\right\|^2 \leq \sigma^2. \tag{4}$$

(3) **Bounded Second Moment**: The second moment of the stochastic gradients is bounded by some $M^2 > 0$, i.e. for all $x \in \mathbb{R}^d$ and agent i:

$$\mathbb{E}\left\|\widetilde{g}_i(x)\right\|^2 \leq M^2. \tag{5}$$

3 The SwarmSGD Algorithm

Algorithm Description. We now describe a decentralized variant of SGD, designed to be executed by a population of n nodes, interacting over the edges of r-regular graph G. We assume that each node i has access to local stochastic gradients $\widetilde{g}^i$, and maintains a model estimate X^i. For simplicity, we will assume that this initial estimate is 0^d at each agent, although its value may be arbitrary. We assume that each agent performs SGD steps on its local estimate X^i. At random times given by a clock of Poisson rate, we pick two neighboring agents i and j uniformly at random from G, and have them average their estimates. More precisely, the interaction can be described as follows:

Algorithm 1 Sequential SwarmSGD pseudocode for each interaction between arbitrary nodes i and j.

```
% Let G be r-regular graph.
% Sample an edge (i, j) of G uniformly at random.
Require: agents i and j chosen for interaction
% choose H_i and H_j
% agent i performs H_i local SGD steps
for q = 1 to H_i do
   X^i ← X^i − ηg̃^i(X^i)
end for
% agent j performs H_j local SGD steps
for q = 1 to H_j do
   X^j ← X^j − ηg̃^j(X^j)
end for
% agents average their estimates coordinate-wise
avg ← (X^i + X^j)/2
X^i ← avg
X^j ← avg
```

Note that, for simplicity, the pseudocode is expressed sequentially, although in practice nodes perform their local SGD steps in parallel. Also, we have assumed a constant learning rate; we will detail the precise update values of the learning rate in the next section.

The main intuition behind the algorithm is that the independent SGD steps will allow nodes to explore local improvements to the objective function on their subset of the data, while the averaging steps provide a decentralized way for the models to remain in sync. In the next section, we will show that, as long as the maximum number of local steps is bounded, this procedure still converges, in

the sense that gradients calculated at average value of models are vanishing as we increase the number of interactions.

4 The Convergence of SwarmSGD

In this section, we analyze the convergence of SwarmSGD algorithm. We consider two versions of Algorithm 1, which differ in the way we choose the number of local SGD steps performed by agents i and j upon interaction.

4.1 Analysis under random interaction times

Fix an integer $H \geq 1$. First, we will consider a version where H_i and H_j are independent geometrically distributed random variables of mean H. Notice that this corresponds to interaction times being chosen by a Poisson clock of constant rate. To handle the fact that the number of local steps upon an interaction is a random variable, in this first case we will require stochastic gradients to satisfy the *bounded second moment assumption*, specified above. Intuitively, this is required since otherwise the "distance travelled" by a node could be virtually unbounded. In this setting, we are able to prove the following theorem:

Theorem 1. *Let f be an non-convex, L-smooth function, whose stochastic gradients satisfy the bounded second moment assumption above. Let the number of local stochastic gradient steps performed by each agent upon interaction be a geometric random variable with mean H. Let the learning rate we use be $\eta = n/\sqrt{T}$. Define $\mu_t = \sum_{i=1}^{n} X_t^i/n$, where X_t^i is a value of model i after t interactions, be the average of the local parameters. Then, for learning rate $\eta = n/\sqrt{T}$ and any number of interactions $T \geq n^4$:*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\mu_t)\|^2 \leq \frac{(f(\mu_0)-f(x^*))}{\sqrt{T}H} + \frac{11H^2\max(1,L^2)M^2}{\sqrt{T}}\max(1, 2r/\lambda_2 + 4r^2/\lambda_2^2).$$

Discussion. We briefly pause to comment on this bound. Notice that each of the upper bound terms has a clear intuitive interpretation: the first represents the reduction in loss relative to the initialization, and gets *divided* by the number of local steps H, since progress is made in this term in every local step; the second represents the influence of the variance of each individual local step multiplied by a term which bounds the impact of the graph topology on the convergence. In particular, this term negatively impacts convergence fo large values of H, L, and M, but gets dampened if the graph is well-connected (corresponding to a large value of λ_2).

Second, importantly, notice that time T in this bound counts the *total* number of interactions. However, in practice $\Theta(n)$ pairwise interactions will occur in parallel. Therefore, it is natural to replace T by nT to obtain speedup

with respect to wall-clock time. At the same time, notice that this speedup is dampened in the second term by the non-trivial additional variance due to noisy local gradient steps, a fact which we will revisit in the experimental section.

Proof Overview. At a high level, the argument rests on two technical ideas. The first idea is to show that, due to the pairwise averaging process, and in spite of the local steps, the nodes' parameters will have to remain concentrated around their mean μ_t. The second is to show that, even though stochastic gradients are taken at *perturbed, noisy* estimates of this mean, the impact of this noise on convergence can be bounded. We note that neither idea is straightforward to implement.

In particular, the main technical difficulty in the proof is to correctly "encode" the fact that parameters are well concentrated around the mean. For this, we define the potential Γ_t, which denotes the variance of models after t interactions. More formally,

$$\Gamma_t = \sum_{i=1}^{n} \|X_t^i - \mu_t\|^2, \tag{6}$$

where $\mu_t = \sum_{i=1}^{n} X_t^i/n$. We bound the expected evolution of Γ_t in terms of r, the degree of nodes in interaction the graph G, and λ_2, the second smallest eigenvalue of the Laplacian of G. For both algorithm variants we consider, our bound depends on the learning rate, number of local steps, and the bound provided by the assumption on the stochastic gradients (the bound M^2 and σ^2). The critical point is that the upper bound on the expectation of Γ_t *does not depend* on the number of interactions t.

The second part of the proof is to use an upper bound on the potential in order to show that $\sum_{i=0}^{T-1} \mathbb{E}[\nabla f(\mu_t)]/T$ is decreasing as we increase the number of interactions. It is well known how to upper bound $\sum_{i=0}^{T-1} \mathbb{E}\|\nabla f(\mu_t)\|^2/T$, in the case when update rule is $\mu_{t+1} = \mu_t - \eta \nabla f(\mu_t)$, $\sum_{i=0}^{T-1} \mathbb{E}[\nabla f(\mu_t)]/T$. In our setting we are able to derive similar bound, by leveraging the fact that deviation of $\nabla f(\mu_t)$ from the update we are actually applying can be upper bounded by variance of the local model values.

Bound on Potential. First we show how to derive an upper bounded on a potential $\Gamma_t = \sum_{i=1}^{n} \|X_t^i - \mu_t\|^2$, if upon interaction between nodes i and j we perform the following steps:

- increase the values of the local models X_t^i and X_t^j by random variables $R_t^i(X_t^i)$ and $R_t^j(X_t^j)$ correspondingly.
- average the values of X_t^i and X_t^j.

The following lemma shows the martingale type bound on the potential:

Lemma 2. *For any time step t , we have:*

$$\mathbb{E}[\Gamma_{t+1}] \leq (1 - \frac{\lambda_2}{rn})\mathbb{E}[\Gamma_t] + (2 + \frac{4r}{\lambda_2})\frac{1}{n}\sum_{i=1}^{n} \mathbb{E}\|R_t^i(X_t^i)\|^2.$$

Here, the $(1 - \frac{\lambda_2}{rn})$ factor in front of $\mathbb{E}[\Gamma_t]$ comes from the fact that we average values of local models after updating them. A roughly similar approach is sometimes used in the analysis of static load balancing schemes, in which each node in a graph starts with a fixed weight value, which they try to balance as well as possible through averaging, see e.g. [7]. Two key elements of novelty in our case are that (1) for us the load balancing process is *dynamic*, in the sense that loads (gradients) get continually added; (2) the load-balancing process we consider is multi-dimensional, whereas usually the literature considers simple scalar weights.

Proof Sketch for Theorem 1. In the case of bounded second moment of stochastic gradients (also random number of local steps), we can unroll the recursion given by Lemma 2. We use the fact that $R_t^i(X_t^i)$ is the sum of H_t^i stochastic gradients multiplied by the learning rate, that $\mathbb{E}[H_t^i] = H$ and that the second moment of each gradient is upper bounded by M^2 in expectation. We get the following time-invariant bound on Γ_t, essentially showing that node models will have bounded variance:

Lemma 3.

$$\mathbb{E}[\Gamma_t] \leq 2n\eta^2 H^2 M^2 (2r/\lambda_2 + 4r^2/\lambda_2^2). \tag{7}$$

With this in place, we apply the L-smoothness of the function f to get:

$$\mathbb{E}[f(\mu_{t+1})] \leq \mathbb{E}[f(\mu_t)] + \mathbb{E}\langle \nabla f(\mu_t), \mu_{t+1} - \mu_t \rangle + \frac{L}{2}\mathbb{E}\|\mu_{t+1} - \mu_t\|^2.$$

Further, we use the fact that $\frac{n}{\eta}(\mu_{t+1} - \mu_t)$ is sum of H stochastic gradients in expectation (up to some constant factor). To upper bound $\frac{L}{2}\mathbb{E}\|\mu_{t+1} - \mu_t\|^2$ we use the second moment bound again. Notice that in the case where the number of local gradient steps is equal to 1, in order to upper bound $\mathbb{E}\langle \nabla f(\mu_t), \mu_{t+1} - \mu_t \rangle$, we can apply the following simple derivation:

$$\begin{aligned}
\mathbb{E}\langle \nabla f(\mu_t), \mu_{t+1} - \mu_t \rangle &= \sum_{i=1}^{n} \frac{2\eta}{n^2} \mathbb{E}\langle \nabla f(\mu_t), -\nabla f(X_t^i) \rangle \\
&= \sum_{i=1}^{n} \frac{2\eta}{n^2} \mathbb{E}\langle \nabla f(\mu_t), \nabla f(\mu_t) - \nabla f(X_t^i) \rangle \\
&\qquad - 2\frac{\eta}{n}\mathbb{E}\|\nabla f(\mu_t)\|^2 \\
&\leq \sum_{i=1}^{n} \frac{2\eta}{n^2} \mathbb{E}\|\nabla f(\mu_t) - \nabla f(X_t^i)\|^2 - \frac{\eta}{n}\mathbb{E}\|\nabla f(\mu_t)\|^2
\end{aligned}$$

We then use L-smoothness one final time to get:

$$\sum_{i=1}^{n} \mathbb{E}\|\nabla f(\mu_t) - \nabla f(X_t^i)\|^2 \leq L^2 \mathbb{E}[\Gamma_t]. \tag{8}$$

The proof of Theorem 1 then follows by carefully re-arranging these terms. The complete argument is presented in the Appendix.

4.2 Analysis under fixed interaction times

In the second variant of the algorithm, we assume that the number of local steps performed by each model when is fixed and is equal to H. In this case we are able to remove the bounded second moment assumption, and are able to prove convergence by just assuming that stochastic gradients have bounded *variance*:

Theorem 4. *Let f be a non-convex, L-smooth function satisfying the bounded variance assumption in Equation 4. Let H be the number of local stochastic gradient steps performed by each agent before each interaction. Define $\mu_t = \sum_{i=1}^{n} X_t^i/n$, where X_t^i is a value of model i after t interactions. For learning rate $\eta = \frac{n}{\sqrt{T}}$ and $T \geq 225n^4H^2\max(1,(2r/\lambda_2 + 4r^2/\lambda_2^2)^2)\max(1,L^2)$ we have that SwarmSGD ensures:*

$$\frac{\sum_{i=0}^{T-1}\mathbb{E}\|\nabla f(\mu_t)\|^2}{T} \leq \frac{1}{\sqrt{T}H}\mathbb{E}[f(\mu_0) - f(\mu_t)] + \frac{28H^2\max(1,L^2)\sigma^2}{\sqrt{T}}\max(1, 2r/\lambda_2 + 4r^2/\lambda_2^2).$$

Discussion. Notice that, relative to Theorem 1, we have the same quadratic dependency on the number of local steps H and on L, but now the second moment bound is replaced by the variance term. Further, this our analysis can be extended to the case where each agent is sampling from its own partition of data, matching the setting assumed by [23], and our implementation. Please see Section F in the Appendix for details.

Proof Sketch for Theorem 4. While the intuitive interpretation of the result is similar, its proof is in fact more subtle, due to the lack of a second-moment bound. Here, our approach is similar to [23], but we are able to handle multiple number of local steps, and handle dynamic pairwise interactions differently. Without the bound in Equation 5, we can only bound the expected sum of potentials up to step T (since we are no longer able to get the bound per step). In particular, we can obtain the following bound on the "average" Γ_t:

Lemma 5. *For $\eta \leq \frac{1}{7HL\sqrt{2r/\lambda_2 + 4r^2/\lambda_2^2}}$, we have that :*

$$\sum_{t=0}^{T}\mathbb{E}[\Gamma_t] \leq \frac{4nr\eta^2\sigma^2H^2T}{\lambda_2}(2 + \frac{4r}{\lambda_2}) + \frac{12nr\eta^2H}{\lambda_2}(2 + \frac{4r}{\lambda_2})\sum_{t=1}^{T}\sum_{q=1}^{H}\mathbb{E}\|\sum_{i=1}^{n}h_i^q(X_t^i)/n\|^2.$$

Here, we define the auxiliary variable $h_i^q(X_t^i)$ as

$$h_i^q(X_t^i) = \mathbb{E}[\widetilde{h}_i^q(X_t^i)] = \nabla f(X_t^i - \sum_{s=0}^{q-1}\eta\widetilde{h}_i^s(X_t^i)),$$

which is the gradient computed after we apply $q-1$ local stochastic gradient steps to the model X_t^i and $\widetilde{h}_i^q(X_t^i)$ is stochastic gradient used at local step q. We can use the L-smoothness of function f as in the proof of the Theorem 1 and after some careful manipulations we get that:

$$\begin{aligned}\mathbb{E}[f(\mu_{t+1})] \leq \mathbb{E}[f(\mu_t)] \\ + \sum_{q=1}^{H}\Bigg(\frac{\eta}{n^2}\sum_{i=1}^{n}\mathbb{E}\|\nabla f(\mu_t) - h_i^q(X_t^i)\|^2 \\ - \frac{\eta}{n}\mathbb{E}\|\nabla f(\mu_t)\|^2 - \mathbb{E}\|\sum_{i=1}^{n} h_i^q(X_t^i)/n\|^2 \\ + \sum_{i=1}^{n}\frac{2LH}{n}\mathbb{E}\|\frac{\eta}{n}\widetilde{h}_i^q(X_t^i)\|^2\Bigg) \end{aligned} \tag{9}$$

Next, we prove some technical lemmas which upper bound each term in the inequality above.

To complete the proof, we sum the inequality 9 from $t=0$ to $t=T-1$ and apply Lemma 5 together with the bounds on the individual terms of the previous inequality. The crucial point is that in the resulting inequality all positive terms with multiplicative factor $\mathbb{E}\|\sum_{i=1}^{n} h_i^q(X_t^i)/n\|^2$ have an η^2 multiplicative factor as well. Hence, we are able to set η small enough and cancel those terms with $-\frac{\eta}{n}\mathbb{E}\|\sum_{i=1}^{n} h_i^q(X_t^i)/n\|^2$. This allows us to complete the proof of Theorem 4 after some careful additional calculation. We refer the reader to the supplementary material for the full argument.

5 Experimental Results

In this section, we validate our analytical results numerically, by applying the algorithm to the computationally-intensive task of training deep neural networks for image classification. For this we map the execution pattern of the algorithm onto a super-computing setting, in which we have a large number of powerful computing nodes, connected by fast communication links. The key overhead in this setting is *synchronization*: at large node counts, the cost of synchronizing all nodes so they execute in lock-step can be very high, see e.g. [22] for numerical results on different workloads. SwarmSGD removes this overhead, since nodes synchronize only sporadically and in pairs.

Target System and Implementation. We run SwarmSGD on the CSCS Piz Daint supercomputer, which is composed of Cray XC50 nodes, each with a Xeon E5-2690v3 CPU and an NVIDIA Tesla P100 GPU, using a state-of-the-art Aries interconnect. Please see [1] for hardware details. We implemented SwarmSGD in Pytorch using MPI one-sided primitives [2], which allow nodes to read eachothers' models for averaging without blocking synchronization. We used SwarmSGD to train ResNets on the classic CIFAR-10 and ImageNet datasets. The code is available as additional material, and can also be executed on standard multi-GPU computing clusters with MPI support.

Model / Dataset	SGD Top-1	SwarmSGD	Parameters
ResNet20 / CIFAR-10	91.5%	91.79%	4 local steps, 35 epochs/model
ResNet18 / ImageNet	69.17%	69.79%	3 local steps, 10 epochs / model
ResNet50 / ImageNet	75.43%	75.68%	2 local steps, 12 epochs / model

Table 1: Parameters for full Top-1 validation accuracy on CIFAR-10 and ImageNet. The step count represents *local SGD steps per model* between two averaging steps, while the number of epochs are counted in terms of data passes per individual model.

Training Process. Our training methodology follows data-parallel training, with some differences due to decentralization, and is identical to previous work on decentralized and local SGD, e.g. [23, 6, 25]. Training proceeds in *epochs*, each of which corresponds to processes collectively performing a full pass over the dataset. At the beginning of each epoch, we shuffle the dataset and partition it among processes.

Ideally, if the algorithm had perfect speedup with respect to the number of iterations per model, each process would need to iterate over the dataset just E_{seq}/n times, where E_{seq} is the number of epochs for sequential SGD. However, as noted in previous work [23, 24, 6] this process is not always able to recover sequential model accuracy within E_{seq}/n epochs. This shortcoming of decentralized methods is justified by Theorems 1 and 4, which say that pairwise averaging and local steps can slow down convergence. Thus, we will allow processes to execute for more epochs, by a constant *multiplier* factor, usually in the interval $[2, 4]$. [6] employ the same procedure, but scale up the total number of epochs by the speedup of their parallel algorithm versus data-parallel SGD. (This usually leads to accurate models, but unfortunately nullifies any speedup gains, since the end-to-end training time will match data-parallel SGD.) References [23, 24] only execute for E_{seq}/n epochs, but lose accuracy relative to the SGD baseline on ImageNet-scale tasks.

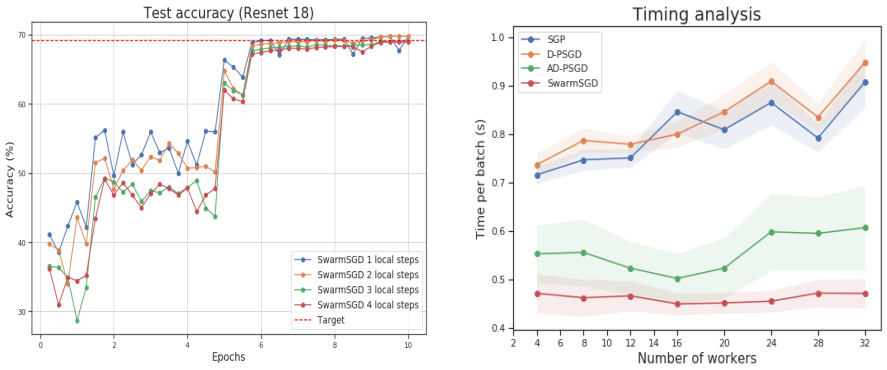
Once we have fixed the number of epochs, *we do not alter the other training hyperparameters*: in particular, the learning rate schedule, momentum and weight decay terms are identical to sequential SGD, for each individual model. Practically, if sequential SGD trains ResNet50 in 90 epochs, decreasing the learning rate at 30 and 60 epochs, then SwarmSGD with 32 nodes and multiplier 4 uses $90 * 4/32 \simeq 12$ epochs per node, decreasing the learning rate at 4 and 8 epochs. (We recover full accuracy at this setting.) Since the synchronization overhead of SwarmSGD is negligible, this ensures non-trivial end-to-end speedup for SwarmSGD.

We now explore the accuracy-scalability trade-off of SwarmSGD, examining the following questions.

Q1: Can SwarmSGD Recover Full Accuracy? This is the first threshold for any distributed training method. In Table 1 provide an overview of parameter values to recover or exceed full accuracy using SwarmSGD. We execute on for

Model / Dataset	1 step	2 steps	3 steps	4 steps
ResNet20 / CIFAR-10	90.84%	91.39%	91.55%	91.79%
ResNet18 / ImageNet	69.71%	69.79%	69.17%	68.99%

Table 2: Validation accuracy on CIFAR-10 and ImageNet versus number of local steps, for $n = 32$. The step count represents *local SGD steps per model* between two pairwise averaging steps. Other parameter values are the same as in the previous experiment.



(a) Convergence of ResNet18/ImageNet versus Local Steps.

(b) Scalability (average time per batch) for SwarmSGD versus previous work. Lower is better.

Figure 1: Convergence and Scalability of SwarmSGD on ImageNet Models.

32 nodes on ImageNet, and 12 nodes on CIFAR-10, with (local) batch size 256. We have investigated local step values between 1 and 5, since we obtain no additional cost savings for higher values. We conclude that SwarmSGD is able to recover full accuracy on these accuracy-sensitive models in spite of local steps and asynchronous averaging, using a relatively small number of updates per model. (On ImageNet, it uses approximately 9x less updates per model relative to the sequential algorithm.) However, its iteration complexity is inferior to centralized methods, since e.g. for ResNet18 the 32 processes collectively execute a total of 32×10 epochs in parallel, relative to only 90 for the sequential model (multiplier value $320/90 \simeq 3.6$).

Q2: Do Local Steps Impact Convergence and Scaling? Next, we focus on the impact of the number of local steps on convergence. Table 2 presents the progression of top-1 validation accuracy versus the number of local steps, under the same parameter values as above, while Figure **??** outlines convergence versus number of epochs relative to a fixed model. We notice that the accuracy is relatively stable when increasing local steps, although the ImageNet experiment confirms the analytical insight that lower number of local steps leads to better convergence. (We note that accuracy on CIFAR-10 decreases slightly at 5 local steps, to 91.44%.) We also investigated the accuracy of the *average* of all

models throughout training. We find that it is usually more accurate than an arbitrary model, but not by significant amounts. This corroborates the claim that individual models tend to stay close to the mean.

In addition, we investigated the impact of the number of local steps on performance. For large models, we find a significant difference between 1 and 2 steps, but the performance benefits trail off for higher values (see also discussion below). We hence do not attempt to optimize for accuracy at high local step values.

Q3: Does SwarmSGD Scale? Yes. The fact that the method should have low synchronization cost relative to node count is not surprising, since it reduces both the frequency and the cost of communication. Figure 1(b) illustrates the communication cost of the algorithm relative to previously proposed decentralized algorithms, in particular D-PSGD [23], AD-PSGD [24], and SGP [6]. The experiment is performed on ResNet50/ImageNet, for local batch size 256, and local step count 2 for SwarmSGD (this setting was used to recover accuracy for the model). The algorithms are fairly implemented and compared in the common framework of SGP [6]. We observe that its scalability behavior of SwarmSGD is superior to that of previous methods. In particular, its cost per batch appears to be almost independent of the number of nodes, which is ideal. Reducing the number of intermediate steps (not shown) reveals that the advantage relative to AD-PSGD comes mainly from the reduction in communication frequency, as expected.

Q4: What are the End-to-end Training Times? Finally, we look at the end-to-end training times are for SwarmSGD relative to the baselines. On 32 nodes, we can use SwarmSGD to train ResNet18 on ImageNet to full accuracy in less than 11 hours, and ResNet50 in less than 22 hours. The data-parallel baseline with global batch size 256 (strong scaling) requires 25.7 hours and 30.2 hours in this setting, respectively. This speedup is quite significant, given that the Aries supercomputing network and the MPI implementation are both state-of-the-art. If we consider weak scaling, that is, large-batch execution with a maximum per-node batch size of 8192 and 4096 respectively, we can reduce the cost of the baseline to 18.23 and 29.7 hours, respectively, at the price of significant additional tuning. Interestingly, SwarmSGD significantly surpasses the performance of both small- and large-batch baselines, with the only hyperparameter being the number of epochs to execute for. Relative to other decentralized methods, SwarmSGD has the advantage of perfect scalability (Figure 1(b)) and accuracy recovery (Table 1). We conclude that SwarmSGD is highly competitive with previous solutions in this setting.

6 Discussion and Future Work

We have analyzed the convergence of SGD in an extremely decoupled model of distributed computing, in which nodes mostly perform independent SGD updates, interspersed with intermittent pairwise averaging steps. We have shown that SGD is able to still converge in this restrictive setting, and moreover can even achieve speedup in the number of nodes in terms of iteration time. The empirical results in a supercomputing environment complement and confirm our

analysis, showing that this method can be competitive with standard solutions. SwarmSGD provides good analytic and empirical properties, even though each model only directly observes a small fraction of the data. Our work opens several avenues for extensions. One natural but non-trivial direction is to study compressed communication; another is to further generalize the bounds in terms of their the assumptions on the objective and on the system setting considered.

References

[1] The CSCS Piz Daint supercomputer. http://www.cscs.ch/computers/piz_daint. Accessed: 2018-1-25.

[2] Mpich: high performance and widely portable implementation of the message passing interface (mpi) standard. http://www.mpich.org/. Accessed: 2018-1-25.

[3] Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. *arXiv preprint arXiv:1704.05021*, 2017.

[4] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Randomized quantization for communication-efficient stochastic gradient descent. In *Proceedings of NIPS 2017*, 2017.

[5] Dana Angluin, James Aspnes, Zoë Diamadi, Michael J Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed computing*, 18(4):235–253, 2006.

[6] Mahmoud Assran, Nicolas Loizou, Nicolas Ballas, and Michael Rabbat. Stochastic gradient push for distributed deep learning. *arXiv preprint arXiv:1811.10792*, 2018.

[7] Petra Berenbrink, Tom Friedetzky, and Zengjian Hu. A new analytical method for parallel, diffusion-type load balancing. *J. Parallel Distrib. Comput.*, 69(1):5461, January 2009.

[8] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[9] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE/ACM Trans. Netw.*, 14(SI):2508–2530, June 2006.

[10] Kai Chen and Qiang Huo. Scalable training of deep learning machines by incremental block training with intra-block parallel optimization and blockwise model-update filtering. In *2016 ieee international conference on acoustics, speech and signal processing (icassp)*, pages 5880–5884. IEEE, 2016.

[11] Nikoli Dryden, Sam Ade Jacobs, Tim Moon, and Brian Van Essen. Communication quantization for data-parallel training of deep neural networks. In *Proceedings of the Workshop on Machine Learning in High Performance Computing Environments*, pages 1–8. IEEE Press, 2016.

[12] John C Duchi, Sorathan Chaturapruek, and Christopher Ré. Asynchronous stochastic convex optimization. *arXiv preprint arXiv:1508.00882*, 2015.

[13] Bhaskar Ghosh and S. Muthukrishnan. Dynamic load balancing by random matchings. *J. Comput. Syst. Sci.*, 53(3):357370, December 1996.

[14] 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. *arXiv preprint arXiv:1706.02677*, 2017.

[15] Demjan Grubic, Leo Tam, Dan Alistarh, and Ce Zhang. Synchronous multi-gpu deep learning with low-precision communication: An experimental study. In *EDBT*, 2018.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[17] Hadrien Hendrikx, Laurent Massoulié, and Francis Bach. Accelerated decentralized optimization with local updates for smooth and strongly convex objectives. *arXiv preprint arXiv:1810.02660*, 2018.

[18] Björn Johansson, Maben Rabi, and Mikael Johansson. A randomized incremental subgradient method for distributed optimization in networked systems. *SIAM Journal on Optimization*, 20(3):1157–1170, 2009.

[19] David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pages 482–491. IEEE, 2003.

[20] Anastasia Koloskova, Sebastian U Stich, and Martin Jaggi. Decentralized stochastic optimization and gossip algorithms with compressed communication. *arXiv preprint arXiv:1902.00340*, 2019.

[21] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-10 dataset. *online: http://www.cs.toronto. edu/kriz/cifar.html*, 55, 2014.

[22] Shigang Li, Tal Ben-Nun, Salvatore Di Girolamo, Dan Alistarh, and Torsten Hoefler. Taming unbalanced training workloads in deep learning with partial collective operations. *arXiv preprint arXiv:1908.04207*, 2019.

[23] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jio Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *arXiv preprint arXiv:1705.09056*, 2017.

[24] Xiangru Lian, Wei Zhang, Ce Zhang, and Ji Liu. Asynchronous decentralized parallel stochastic gradient descent. *arXiv preprint arXiv:1710.06952*, 2017.

[25] Tao Lin, Sebastian U Stich, Kumar Kshitij Patel, and Martin Jaggi. Don't use large mini-batches, use local sgd. *arXiv preprint arXiv:1808.07217*, 2018.

[26] Angelia Nedic, Alex Olshevsky, and Wei Shi. Achieving geometric convergence for distributed optimization over time-varying graphs. *SIAM Journal on Optimization*, 27(4):2597–2633, 2017.

[27] Angelia Nedic and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48, 2009.

[28] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*, pages 693–701, 2011.

[29] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[30] C. M. De Sa, C. Zhang, K. Olukotun, and C. Re. Taming the wild: A unified analysis of hogwild-style algorihms. In *Advances in Neural Information Processing Systems*, 2015.

[31] F. Seide, H. Fu, L. G. Jasha, and D. Yu. 1-bit stochastic gradient descent and application to data-parallel distributed training of speech dnns. *Interspeech*, 2014.

[32] Ohad Shamir and Nathan Srebro. Distributed stochastic optimization and learning. In *2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 850–857. IEEE, 2014.

[33] Sebastian U Stich. Local sgd converges fast and communicates little. *arXiv preprint arXiv:1805.09767*, 2018.

[34] Nikko Strom. Scalable distributed dnn training using commodity gpu cloud computing. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[35] Hanlin Tang, Ce Zhang, Shaoduo Gan, Tong Zhang, and Ji Liu. Decentralization meets quantization. *CoRR, abs/1803.06443*, 2018.

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

[37] 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 *Advances in Neural Information Processing Systems*, pages 1508–1518, 2017.

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

[39] Yang You, Igor Gitman, and Boris Ginsburg. Scaling sgd batch size to 32k for imagenet training. *arXiv preprint arXiv:1708.03888*, 2017.

A Summary of the Appendix sections

Appendix contains the following sections:

- In **Section B** we compare SwarmSGD with some of the existing algorithms. We list convergence bounds and the assumptions needed to achieve them.
- In **Section C** we prove the crucial lemma which derives martingale type bound on the potential Γ (See Lemma 2), in the case when increment of model i with value X_t^i at step t, is a random variable $R_t^i(X_t^i)$.
- In **Section D** we provide the full proof of the Theorem 1, which shows the convergence of SwarmSGD assuming the second moment bound on the gradients. Recall that the number of local steps in this case is a geometric random variable with mean H.
- In **Section E** we prove the Theorem 4. In this case we do not assume the second moment bound and the number of the local steps performed by each agent is a fixed number H.
- In **Section F** we show that SwarmSGD analysis can be extended to the case where each agent is sampling from its own partition of data.
- In **Section G** we provide additional experimental results for SwarmSGD.

B Comparison of results

In this section we compare convergence rates of existing algorithms, while specifying the bounds they require for convergence. In the tables T-corresponds to the parallel time and n is a number of processes. We use the following notations for needed bounds(or assumptions):

(1) $\boldsymbol{\sigma^2}$ - bound on the variance of gradient.

(2) $\mathbf{M^2}$ - bound on the second moment of gradient.

(3) **PL** - Polyak-ojasiewicz assumption.

(4) **d** - bounded dimension.

(5) $\boldsymbol{\lambda_2}$ - bounded spectral gap of the averaging matrix(interaction graph in case of SwarmSGD).

(6) $\boldsymbol{\tau}$ - bounded message delay.

(7) **r** - interaction graph is r-regular.

(8) $\boldsymbol{\Delta}$ - bounded diameter of interaction graph.

	Global synchronization	Assumptions	Convergance Rate
SwarmSGD	NO	σ^2, λ_2, r	$O(1/\sqrt{Tn})$
SwarmSGD	NO	M^2, λ_2, r	$O(1/\sqrt{Tn})$
AD-PSGD [24]	NO	$\sigma^2, \lambda_2, \tau$	$O(1/\sqrt{Tn})$
SGP [6]	NO	$\sigma^2, d, \Delta, \tau$	$O(1/\sqrt{Tn})$

Table 3: **non-convex case**

C Potential Bound

We are given a simple undirected graph G, with n nodes(for convenience we number them from 1 to n) and edge set E. Each node is adjacent to exactly r nodes. Let $\mathcal{L}$ be the Laplacian matrix of G and let let λ_2 be a second smallest eigenvalue of $\mathcal{L}$.

First we state the following lemma from [13]

Lemma 6.

$$\lambda_2 = \min_{v=(v_1,v_2,\ldots,v_n)} \left\{ \frac{v^T \mathcal{L} v}{v^T v} | \sum_{i=1}^{n} v_i = 0 \right\} \tag{10}$$

Next, we define a process on the graph G. Each node i of graph G keeps the vector model $X_t^i \in \mathbb{R}^d$, where t is the number of interactions. Interaction is defined as follows: we pick edge $e = (u, v)$ of G uniformly at random and update the vector models correspondingly. We set $X_{t+1}^u = X_{t+1}^v = (X_t^u + X_t^v + R_t^u(X_t^u) + R_t^v(X_t^v))/2$ and $X_{t+1}^w = X_t^w$ for $w \neq u, v$. Here, for time step t and node u, R_t^u is a random function from $\mathbb{R}^d$ to $\mathbb{R}^d$. Lastly, let $\mu_t = \sum_{i=1}^n X_t^i/n$ be the average of models at step t and let $\Gamma_t = \sum_{i=1}^n \|X_t^i - \mu_t\|^2$ be a potential at time step t.

Now, we show that Lemma 6 can be used to lower bound $\sum_{(i,j)\in E} \|X_t^i - X_t^j\|^2$:

Lemma 7.

$$\sum_{(i,j)\in E} \|X_t^i - X_t^j\|^2 \geq \lambda_2 \sum_{i=1}^{n} \|X_t^i - \mu_t\|^2 = \lambda_2 \Gamma_t. \tag{11}$$

Proof. Observe that

$$\sum_{(i,j)\in E} \|X_t^i - X_t^j\|^2 = \sum_{(i,j)\in E} \|(X_t^i - \mu_t) - (X_t^j - \mu_t)\|^2. \tag{12}$$

Also, notice that Lemma 6 means that for every vector $v = (v_!, v_2, ..., v_n)$ such that $\sum_{i=1}^n v_i = 0$, we have :

$$\sum_{(i,j)\in E} (v_i - v_j)^2 \geq \lambda_2 \sum_{i=1}^{n} v_i^2. \tag{13}$$

□

Since $\sum_{i=1}^{n}(X_t^i - \mu_t)$ is a 0 vector, we can apply the above inequality to the each of d components of the vectors
$X_t^1 - \mu_t, X_t^2 - \mu_t, ..., X_t^n - \mu_t$ separately and by properties of two norm we get the proof of the lemma.

We proceed by proving the following lemma which upper bounds the expected change in potential:

Lemma 2. *For any time step t , we have:*

$$\mathbb{E}[\Gamma_{t+1}] \leq (1 - \frac{\lambda_2}{rn})\mathbb{E}[\Gamma_t] + (2 + \frac{4r}{\lambda_2})\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\|R_t^i(X_t^i)\|^2.$$

Proof. First we bound change in potential $\Delta_t = \Gamma_{t+1} - \Gamma_t$ for some time step $t > 0$. Let $\Delta_t^{i,j}$ be a change in potential when we choose agents i and j for interaction. We get that

$$\mathbb{E}\Big[\Delta_t | X_t\Big] = \sum_{(i,j)\in E}\frac{1}{rn/2}\mathbb{E}\Big[\Delta_t^{i,j}|X_t\Big]. \tag{14}$$

For $(i,j) \in E$ we have that:

$$X_{t+1}^i - \mu_{t+1} = X_{t+1}^j - \mu_{t+1} = (X_t^i + X_t^j)/2 + \frac{n-2}{2n}(R_t^i(X_t^i) + R_t^j(X_t^j)) - \mu_t.$$

For $k \neq i, j$ we get that

$$X_{t+1}^k - \mu_{t+1} = X_t^k - \frac{1}{n}(R_t^i(X_t^i) + R_t^j(X_t^j)) - \mu_t.$$

This gives us that

$$\begin{aligned}
\mathbb{E}\Big[\Delta_t^{i,j}|X_t\Big] = \;& \mathbb{E}\Big\|(X_t^i + X_t^j)/2 + \frac{n-2}{2n}(R_t^i(X_t^i) + R_t^j(X_t^j)) - \mu_t\|^2 - \|X_t^i - \mu_t\Big\|^2 \\
& + \mathbb{E}\Big\|(X_t^i + X_t^j)/2 + \frac{n-2}{2n}(R_t^i(X_t^i) + R_t^j(X_t^j)) - \mu_t\|^2 - \|X_t^j - \mu_t\Big\|^2 \\
& + \sum_{k\neq i,j}\Big(\mathbb{E}\Big\|X_t^k - \frac{1}{n}(R_t^i(X_t^i) + R_t^j(X_t^j)) - \mu_t\Big\|^2 - \|X_t^k - \mu_t\|^2\Big) \\
= \;& 2\|(X_t^i - \mu_t)/2 + (X_t^j - \mu_t)/2\|^2 - \|X_t^i - \mu_t\|^2 - \|X_t^j - \mu_t\|^2 \\
& + \frac{n-2}{n}\mathbb{E}\langle R_t^i(X_t^i) + R_t^j(X_t^j), (X_t^i - \mu_t) + (X_t^j - \mu_t)\rangle \\
& + 2\Big(\frac{n-2}{2n}\Big)^2\mathbb{E}\|R_t^i(X_t^i) + R_t^j(X_t^j)\|^2 \\
& + \sum_{k\neq i,j}\Big(-\frac{2}{n}\mathbb{E}\langle R_t^i(X_t^i) + R_t^j(X_t^j), X_t^k - \mu_t\rangle + \frac{1}{n^2}\mathbb{E}\|R_t^i(X_t^i) + R_t^j(X_t^j)\|^2\Big)
\end{aligned}$$

Observe that

$$\sum_{k=1}^{n} \mathbb{E}\langle R_t^i(X_t^i) + R_t^j(X_t^j), X_t^k - \mu_t \rangle = 0.$$

Hence:

$$\begin{aligned}
\mathbb{E}\Big[\Delta_t^{i,j} | X_t\Big] \leq \quad & 2\|(X_t^i - \mu_t)/2 + (X_t^j - \mu_t)/2\|^2 - \|X_t^i - \mu_t\|^2 - \|X_t^j - \mu_t\|^2 \\
& + \mathbb{E}\langle R_t^i(X_t^i) + R_t^j(X_t^j), (X_t^i - \mu_t) + (X_t^j - \mu_t) \rangle \\
& + 4\Big(\frac{n-2}{2n}\Big)^2 (\mathbb{E}\|R_t^i(X_t^i)\|^2 + \mathbb{E}\|R_t^j(X_t^j)\|^2) + \sum_{k \neq i,j} \frac{2}{n^2} (\mathbb{E}\|R_t^i(X_t^i)\|^2 + \mathbb{E}\|R_t^j(X_t^j)\|^2) \\
\leq \quad & - \|X_t^i - X_t^j\|^2/2 \\
& + \mathbb{E}\langle R_t^i(X_t^i) + R_t^j(X_t^j), (X_t^i - \mu_t) + (X_t^j - \mu_t) \rangle \\
& + (\mathbb{E}\|R_t^i(X_t^i)\|^2 + \mathbb{E}\|R_t^j(X_t^j)\|^2)
\end{aligned}$$

This gives us :

$$\begin{aligned}
\mathbb{E}[\Delta_t | X_t] \leq & \sum_{(i,j) \in E} \frac{1}{rn/2} \Big(- \|X_t^i - X_t^j\|^2/2 + \mathbb{E}\langle R_t^i(X_t^i) + R_t^j(X_t^j), (X_t^i - \mu_t) + (X_t^j - \mu_t) \rangle \\
& \qquad + \mathbb{E}\|R_t^i(X_t^i)\|^2 + \mathbb{E}\|R_t^j(X_t^j)\|^2 \Big) \\
\overset{Lemma 7}{\leq} & -\frac{\lambda_2}{rn/2} \Gamma_t + \frac{2}{n} \sum_{i=1}^{n} \mathbb{E}\|R_t^i(X_t^i)\|^2 \\
& + \sum_{(i,j) \in E} \frac{1}{rn/2} \mathbb{E}\langle R_t^i(X_t^i) + R_t^j(X_t^j), (X_t^i - \mu_t) + (X_t^j - \mu_t) \rangle.
\end{aligned}$$

Further, we have that

$$\begin{aligned}
& \sum_{(i,j) \in E} \frac{1}{rn/2} \mathbb{E}\langle R_t^i(X_t^i) + R_t^j(X_t^j), (X_t^i - \mu_t) + (X_t^j - \mu_t) \rangle \\
& \leq \sum_{(i,j) \in E} \frac{1}{rn} \Big(\frac{2r}{\lambda_2} \mathbb{E}\|R_t^i(X_t^i) + R_t^j(X_t^j)\|^2 + \frac{\lambda_2}{2r} \mathbb{E}\|(X_t^i - \mu_t) + (X_t^j - \mu_t)\|^2 \Big) \\
& \leq \sum_{(i,j) \in E} \frac{1}{rn} \Big(\frac{4r}{\lambda_2} (\mathbb{E}\|R_t^i(X_t^i)\|^2 + \mathbb{E}\|R_t^j(X_t^j)\|^2) + \frac{\lambda_2}{r} (\mathbb{E}\|X_t^i - \mu_t\|^2 + \mathbb{E}\|X_t^j - \mu_t\|^2) \Big) \\
& = \frac{\lambda_2}{rn} \Gamma_t + \frac{4r}{\lambda_2 n} \sum_{i=1}^{n} \mathbb{E}\|R_t^i(X_t^i)\|^2
\end{aligned}$$

By combining the above two inequalities we get that

$$\mathbb{E}[\Delta_t | X_t] \leq -\frac{\lambda_2}{rn} \Gamma_t + (2 + \frac{4r}{\lambda_2}) \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}\|R_t^i(X_t^i)\|^2.$$

Hence, considering the definition of Δ_t and the fact that $\mathbb{E}[\Gamma_{t+1}] = \mathbb{E}[\mathbb{E}[\Gamma_{t+1} | X_t]]$ we get the proof of the Lemma.

□

D Analysis with second Moment Bound

In this section we address the case when function we want to optimize is non-convex by using constant learning rate over all iterations and process. We start by inductively defining the decrements we apply to the model with the value X_t^i locally, without taking the multiplicative learning rate factor into the account. Let H_t^i be a geometric random variable with a mean H. H_t^i is the expected number of local steps agent i performs if it interacts at step $t+1$.

$$\widetilde{h}_i^0(X_t^i) = 0.$$

and for $1 \leq q \leq H_t^i$ let:

$$\widetilde{h}_i^q(X_t^i) = \widetilde{g}_i(X_t^i - \sum_{s=0}^{q-1} \eta \widetilde{h}_i^s(X_t^i)).$$

Also , for $1 \leq q \leq H_t^i$, let

$$h_i^q(X_t^i) = \nabla f(X_t^i - \sum_{s=0}^{q-1} \eta \widetilde{h}_i^s(X_t^i)).$$

be the expected value of $\widetilde{h}_i^q(X_t^i)$ taken over the randomness of the stochastic gradient $\widetilde{g}_i$. Let $\widetilde{h}_i(X_t^i)$ be the total amount by which we decrement the model(again without the learning rate factor):

$$\widetilde{h}_i(X_t^i) = \sum_{q=1}^{H_t^i} \widetilde{h}_i^q(X_t^i).$$

The new value of the model X_t^i after local update is(before it gets averaged with the other model)

$$X_t^i - \eta \widetilde{h}_i(X_t^i) = X_t^i - \eta \sum_{q=1}^{H_t^i} \widetilde{h}_i^q(X_t^i) = X_t^i - \eta \sum_{q=1}^{H_t^i} \widetilde{g}_i(X_t^i - \sum_{s=0}^{q-1} \eta \widetilde{h}_i^s(X_t^i))$$

Lemma 8.

$$\sum_{i=1}^{n} \mathbb{E}\|\eta \widetilde{h}_i(X_t^i)\|^2 \leq 2nH^2M^2.$$

Proof.

$$\begin{aligned}
\sum_{i=1}^{n} \mathbb{E}\|\eta \widetilde{h}_i(X_t^i)\|^2 &= \sum_{u=1}^{\infty} Pr[H_t^i = u] \sum_{i=1}^{n} \mathbb{E}\| \sum_{q=1}^{u} \widetilde{h}_i^q(X_t^i)\|^2 \\
&\leq \sum_{u=1}^{\infty} Pr[H_t^i = u] \sum_{i=1}^{n} u \sum_{q=1}^{u} \mathbb{E}\|\widetilde{h}_i^q(X_t^i)\|^2 \\
&\leq \sum_{u=1}^{\infty} Pr[H_t^i = u] u^2 \sum_{i=1}^{n} M^2 \leq 2nH^2M^2.
\end{aligned}$$

Where in the last step we used

$$\sum_{u=1}^{\infty} Pr[H_t^i = u]u^2 = \mathbb{E}[(H_t^i)^2] = 2H^2 - H \leq 2H^2.$$

□

By applying Lemma 2 with $R_t^i(X_t^i) = -\eta\widetilde{h}_i(X_t^i)$, we get the following lemma

Lemma 9.

$$\mathbb{E}[\Gamma_{t+1}] \leq (1 - \frac{\lambda_2}{rn})\mathbb{E}[\Gamma_t] + 2\eta^2(2 + \frac{4r}{\lambda_2})H^2M^2.$$

Proof.

$$\begin{aligned}\mathbb{E}[\Gamma_{t+1}] &\overset{Lemma2}{\leq} (1 - \frac{\lambda_2}{rn})\mathbb{E}[\Gamma_t] + (2 + \frac{4r}{\lambda_2})\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\| - \eta\widetilde{h}_i(X_t^i)\|^2 \\ &\overset{Lemma8}{\leq} (1 - \frac{\lambda_2}{rn})\mathbb{E}[\Gamma_t] + 2\eta^2(2 + \frac{4r}{\lambda_2})H^2M^2.\end{aligned}$$

□

The lemma above allows us to bound the expected potential at step t:

Lemma 3.

$$\mathbb{E}[\Gamma_t] \leq 2n\eta^2H^2M^2(2r/\lambda_2 + 4r^2/\lambda_2^2). \tag{17}$$

Proof. We prove by using induction. Base case $t = 0$ trivially holds. For an induction step step we assume that
$\mathbb{E}[\Gamma_t] \leq n\eta^2H^2M^2(2r/\lambda_2 + 4r^2/\lambda_2^2)$. We get that :

$$\begin{aligned}\mathbb{E}[\Gamma_{t+1}] &\leq (1 - \frac{\lambda_2}{rn})\mathbb{E}[\Gamma_t] + 2\eta^2(2 + \frac{4r}{\lambda_2})H^2M^2 \\ &\leq (1 - \frac{\lambda_2}{rn})2n\eta^2H^2M^2(2r/\lambda_2 + 4r^2/\lambda_2^2) + 2\eta^2(2 + \frac{4r}{\lambda_2})H^2M^2 \\ &= 2n\eta^2H^2M^2(2r/\lambda_2 + 4r^2/\lambda_2^2).\end{aligned}$$

□

Lemma 10.

$$\sum_{i=1}^{n}\mathbb{E}\langle\nabla f(\mu_t), -\widetilde{h}_i(X_t^i)\rangle \leq HL^2\mathbb{E}[\Gamma_t] - \frac{Hn}{2}\mathbb{E}\|\nabla f(\mu_t)\|^2 + 6H^3nL^2M^2\eta^2. \tag{18}$$

Proof.

$$
\begin{aligned}
\sum_{i=1}^{n}\mathbb{E}\langle\nabla f(\mu_t),-\widetilde{h}_i(X_t^i)\rangle &= \sum_{i=1}^{n}\sum_{u=1}^{\infty} Pr[H_t^i=u]\mathbb{E}\langle\nabla f(\mu_t),-\sum_{q=1}^{u}\widetilde{h}_i^q(X_t^i)\rangle \\
&= \sum_{i=1}^{n}\sum_{u=1}^{\infty} Pr[H_t^i=u]\sum_{q=1}^{u}\Big(\mathbb{E}\langle\nabla f(\mu_t),\nabla f(\mu_t)-h_i^q(X_t^i)\rangle-\mathbb{E}\|\nabla f(\mu_t)\|^2\Big) \\
&= \sum_{i=1}^{n}\sum_{u=1}^{\infty} Pr[H_t^i=u]\sum_{q=1}^{u}\Big(\mathbb{E}\langle\nabla f(\mu_t),\nabla f(\mu_t)-\nabla f(X_t^i-\sum_{s=0}^{q-1}\eta\widetilde{h}_i^s(X_t^i))\rangle-\mathbb{E}\|\nabla f(\mu_t)\|^2\Big) \\
&= \sum_{i=1}^{n}\sum_{u=1}^{\infty} Pr[H_t^i=u]\sum_{q=1}^{u}\Big(\mathbb{E}\|\nabla f(\mu_t)-\nabla f(X_t^i-\sum_{s=0}^{q-1}\eta\widetilde{h}_i^s(X_t^i))\|^2/2-\mathbb{E}\|\nabla f(\mu_t)\|^2/2\Big) \\
&\overset{L-smoothness}{\leq} \sum_{i=1}^{n}\sum_{u=1}^{\infty} Pr[H_t^i=u]\sum_{q=1}^{u}\Big(L^2\mathbb{E}\|\mu_t-X_t^i+\sum_{s=0}^{q-1}\eta\widetilde{h}_i^s(X_t^i))\|^2/2-\mathbb{E}\|\nabla f(\mu_t)\|^2/2\Big) \\
&\leq \sum_{i=1}^{n}\sum_{u=1}^{\infty} Pr[H_t^i=u]\sum_{q=1}^{u}\Big(L^2\mathbb{E}\|\mu_t-X_t^i\|^2+L^2\mathbb{E}\|\sum_{s=0}^{q-1}\eta\widetilde{h}_i^s(X_t^i))\|^2-\mathbb{E}\|\nabla f(\mu_t)\|^2/2\Big) \\
&\leq \sum_{i=1}^{n}\sum_{u=1}^{\infty} Pr[H_t^i=u]\sum_{q=1}^{u}\Big(L^2\mathbb{E}\|\mu_t-X_t^i\|^2+L^2\eta^2q\sum_{s=0}^{q-1}\mathbb{E}\|\widetilde{h}_i^s(X_t^i))\|^2-\mathbb{E}\|\nabla f(\mu_t)\|^2/2\Big) \\
&\leq \sum_{i=1}^{n}\sum_{u=1}^{\infty} Pr[H_t^i=u]\sum_{q=1}^{u}\Big(L^2\mathbb{E}\|\mu_t-X_t^i\|^2+L^2\eta^2q^2M^2-\mathbb{E}\|\nabla f(\mu_t)\|^2/2\Big) \\
&= \sum_{i=1}^{n}\sum_{u=1}^{\infty} Pr[H_t^i=u]u\Big(L^2\mathbb{E}\|\mu_t-X_t^i\|^2-\mathbb{E}\|\nabla f(\mu_t)\|^2/2\Big) \\
&\qquad + \sum_{i=1}^{n}\sum_{u=1}^{\infty} Pr[H_t^i=u]u(u+1)(2u+1)L^2M^2\eta^2/6 \\
&= HL^2\mathbb{E}[\Gamma_t]-\frac{Hn}{2}\mathbb{E}\|\nabla f(\mu_t)\|^2+\sum_{i=1}^{n}\sum_{u=1}^{\infty} Pr[H_t^i=u]u(u+1)(2u+1)L^2M^2\eta^2/6 \\
&\leq HL^2\mathbb{E}[\Gamma_t]-\frac{Hn}{2}\mathbb{E}\|\nabla f(\mu_t)\|^2+\sum_{i=1}^{n}\sum_{u=1}^{\infty} Pr[H_t^i=u]u^3L^2M^2\eta^2 \\
&\leq \sum_{i=1}^{n}\sum_{u=1}^{\infty} Pr[H_t^i=u]u(u+1)(2u+1)L^2M^2\eta^2/6 \\
&\leq HL^2\mathbb{E}[\Gamma_t]-\frac{Hn}{2}\mathbb{E}\|\nabla f(\mu_t)\|^2+\sum_{i=1}^{n}\sum_{u=1}^{\infty} Pr[H_t^i=u]u^3L^2M^2\eta^2 \\
&\leq HL^2\mathbb{E}[\Gamma_t]-\frac{Hn}{2}\mathbb{E}\|\nabla f(\mu_t)\|^2+6H^3nL^2M^2\eta^2.
\end{aligned}
$$

Where in the last step we used:

$$\sum_{u=1}^{\infty} Pr[H_t^i = u]u^3 = \mathbb{E}[(H_t^i)^3] = 6H^3 - 6H^2 + H \leq 6H^3.$$

□

Theorem 1. *Let f be an non-convex, L-smooth, function satisfying assumption 5, whose minimum $x^\star$ we are trying to find via the SwarmSGD procedure given in Algorithm 1. Let the number of local stochastic gradient steps performed by each agent upon interaction be a geometric random variable with mean H. Let the learning rate we use be $\eta = n/\sqrt{T}$. Define $\mu_t = \sum_{i=1}^{n} X_t^i/n$, where X_t^i is a value of model i after t interactions. Then, for learning rate $\eta = n/\sqrt{T}$ and any $T \geq n^4$:*

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\|\nabla f(\mu_t)\|^2 \leq \frac{(f(\mu_0) - f(x^*))}{\sqrt{T}H} + \frac{11H^2 \max(1, L^2)M^2}{\sqrt{T}} \max(1, 2r/\lambda_2 + 4r^2/\lambda_2^2).$$

Proof.

$$\mathbb{E}[f(\mu_{t+1})] \overset{L-smoothness}{\leq} \mathbb{E}[f(\mu_t)] + \mathbb{E}\langle \nabla f(\mu_t), \mu_{t+1} - \mu_t \rangle + \frac{L}{2}\mathbb{E}\|\mu_{t+1} - \mu_t\|^2 \tag{19}$$

$$= \mathbb{E}[f(\mu_t)] + \sum_{(i,j)\in E} \frac{1}{rn/2}\mathbb{E}\langle \nabla f(\mu_t), -\frac{\eta}{n}\widetilde{h}_i(X_t^i) - \frac{\eta}{n}\widetilde{h}_j(X_t^j)\rangle \tag{20}$$

$$+ \sum_{(i,j)\in E} \frac{L}{rn}\mathbb{E}\|\frac{\eta}{n}\widetilde{h}_i(X_t^i) + \frac{\eta}{n}\widetilde{h}_j(X_t^j)\|^2 \tag{21}$$

$$\leq \mathbb{E}[f(\mu_t)] + \sum_{(i,j)\in E} \frac{1}{rn/2}\mathbb{E}\langle \nabla f(\mu_t), -\frac{\eta}{n}\widetilde{h}_i(X_t^i) - \frac{\eta}{n}\widetilde{h}_j(X_t^j)\rangle$$

$$+ \sum_{(i,j)\in E} \frac{2L}{rn}\mathbb{E}\Big[\|\frac{\eta}{n}\widetilde{h}_i(X_t^i)\|^2 + \|\frac{\eta}{n}\widetilde{h}_j(X_t^j)\|^2\Big] \tag{22}$$

$$= \mathbb{E}[f(\mu_t)] + \sum_{i=1}^{n} \frac{2}{n}\mathbb{E}\langle \nabla f(\mu_t), -\frac{\eta}{n}\widetilde{h}_i(X_t^i)\rangle + \sum_{i=1}^{n} \frac{2L}{n}\mathbb{E}\|\frac{\eta}{n}\widetilde{h}_i(X_t^i)\|^2 \tag{23}$$

$$\overset{Lemma10}{\leq} \mathbb{E}[f(\mu_t)] + \frac{2\eta}{n^2}\Big(HL^2\mathbb{E}[\Gamma_t] - \frac{Hn}{2}\mathbb{E}\|\nabla f(\mu_t)\|^2 + 6H^3nL^2M^2\eta^2\Big) \tag{24}$$

$$+ \sum_{i=1}^{n} \frac{2L\eta^2}{n^3}\mathbb{E}\|\widetilde{h}_i(X_t^i)\|^2 \tag{25}$$

$$\overset{Lemma8}{\leq} \mathbb{E}[f(\mu_t)] + \frac{2\eta}{n^2}\Big(HL^2\mathbb{E}[\Gamma_t] - \frac{Hn}{2}\mathbb{E}\|\nabla f(\mu_t)\|^2 + 6H^3nL^2M^2\eta^2\Big) \tag{26}$$

$$+ \frac{4L\eta^2H^2M^2}{n^2} \tag{27}$$

recall that by Lemma 3 we have that $\mathbb{E}[\Gamma_t] \leq 2n\eta^2H^2M^2(2r/\lambda_2 + 4r^2/\lambda_2^2)$, hence the above inequality becomes:

$$\mathbb{E}[f(\mu_{t+1})] - \mathbb{E}[f(\mu_t)] \leq \frac{4L^2\eta^3M^2H^3}{n}(2r/\lambda_2 + 4r^2/\lambda_2^2) - \frac{\eta H}{n}\mathbb{E}\|\nabla f(\mu_t)\|^2 \tag{28}$$

$$+ \frac{4L\eta^2H^2M^2}{n^2} + \frac{6H^3L^2M^2\eta^3}{n}. \tag{29}$$

by summing the above inequality for $t = 0$ to $t = T - 1$, we get that

$$\mathbb{E}[f(\mu_T)] - f(\mu_0) \leq \sum_{t=0}^{T-1}\Big(\frac{4L^2\eta^3M^2H^3}{n}(2r/\lambda_2 + 4r^2/\lambda_2^2) - \frac{\eta H}{n}\mathbb{E}\|\nabla f(\mu_t)\|^2$$

$$+ \frac{4L\eta^2H^2M^2}{n^2} + \frac{6H^3L^2M^2\eta^3}{n}\Big).$$

From this we get that :

$$\sum_{t=0}^{T-1} \frac{\eta H}{n} \mathbb{E}\|\nabla f(\mu_t)\|^2 \leq f(\mu_0) - \mathbb{E}[f(\mu_T)] + \frac{4TL^2\eta^3 M^2 H^3}{n}(2r/\lambda_2 + 4r^2/\lambda_2^2) + \frac{4TL\eta^2 H^2 M^2}{n^2} + \frac{6TH^3L^2M^2\eta^3}{n}. \tag{30}$$

Note that $\mathbb{E}[f(\mu_T)] \geq f(x^*)$, hence after multiplying the above inequality by $\frac{n}{\eta HT}$ we get that

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\|\nabla f(\mu_t)\|^2 \leq \frac{n(f(\mu_0) - f(x^*))}{TH\eta} + L^2\eta^2 M^2 H^2(2r/\lambda_2 + 4r^2/\lambda_2^2) + \frac{4L\eta H M^2}{n} + 6H^2L^2M^2\eta^2.$$

Observe that $\eta = n/\sqrt{T} \leq 1/n$, since $T \geq n^4$. This allows us to finish the proof:

$$\begin{aligned}
\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\|\nabla f(\mu_t)\|^2 &\leq \frac{n(f(\mu_0) - f(x^*))}{TH\eta} + \frac{L^2\eta M^2 H^2}{n}(2r/\lambda_2 + 4r^2/\lambda_2^2) + \frac{4L\eta H M^2}{n} + \frac{6H^2L^2M^2\eta}{n} \\
&= \frac{(f(\mu_0) - f(x^*))}{\sqrt{T}H} + \frac{L^2M^2H^2}{\sqrt{T}}(2r/\lambda_2 + 4r^2/\lambda_2^2) + \frac{4LHM^2}{\sqrt{T}} + \frac{6H^2L^2M^2}{\sqrt{T}} \\
&\leq \frac{(f(\mu_0) - f(x^*))}{\sqrt{T}H} + \frac{11H^2\max(1, L^2)M^2}{\sqrt{T}}\max(1, 2r/\lambda_2 + 4r^2/\lambda_2^2).
\end{aligned}$$

□

E Analysis without Second Moment Bound

We start by inductively defining the decrements we apply to the model with the value X_t^i locally, without taking the multiplicative learning rate factor into the account. Let

$$\widetilde{h}_i^0(X_t^i) = 0.$$

and for $1 \leq q \leq H$ let:

$$\widetilde{h}_i^q(X_t^i) = \widetilde{g}_i(X_t^i - \sum_{s=0}^{q-1} \eta \widetilde{h}_i^s(X_t^i)).$$

Also , for $1 \leq q \leq H$, let

$$h_i^q(X_t^i) = \nabla f(X_t^i - \sum_{s=0}^{q-1} \eta \widetilde{h}_i^s(X_t^i)).$$

be the expected value of $\widetilde{h}_i^q(X_t^i)$ taken over the randomness of the stochastic gradient $\widetilde{g}_i$. Let $\widetilde{h}_i(X_t^i)$ be the total amount by which we decrement the model(again without the learning rate factor):

$$\widetilde{h}_i(X_t^i) = \sum_{q=1}^{H} \widetilde{h}_i^q(X_t^i).$$

The new value of the model X_t^i after local update is(before it gets averaged with the other model)

$$X_t^i - \eta\widetilde{h}_i(X_t^i) = X_t^i - \eta\sum_{q=1}^{H}\widetilde{h}_i^q(X_t^i) = X_t^i - \eta\sum_{q=1}^{H}\widetilde{g}_i(X_t^i - \sum_{s=0}^{q-1}\eta\widetilde{h}_i^s(X_t^i))$$

By applying Lemma 2 with $R_t^i(X_t^i) = -\eta\widetilde{h}_i(X_t^i)$, we get the following lemma

Lemma 11.

$$\mathbb{E}[\Gamma_{t+1}] \leq (1-\frac{\lambda_2}{rn})\mathbb{E}[\Gamma_t] + \frac{\eta^2 H}{n}(2+\frac{4r}{\lambda_2})\sum_{q=1}^{H}\sum_{i=1}^{n}\mathbb{E}\|\widetilde{h}_i^q(X_t^i)\|^2.$$

Proof.

$$\mathbb{E}[\Gamma_{t+1}] \overset{Lemma2}{\leq} (1-\frac{\lambda_2}{rn})\mathbb{E}[\Gamma_t] + (2+\frac{4r}{\lambda_2})\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\| - \eta\widetilde{h}_i(X_t^i)\|^2 \leq (1-\frac{\lambda_2}{rn})\mathbb{E}[\Gamma_t] + \frac{\eta^2}{n}(2+\frac{4r}{\lambda_2})\sum_{i=1}^{n}\mathbb{E}\|\widetilde{h}_i(X_t^i)\|^2$$
$$\leq (1-\frac{\lambda_2}{rn})\mathbb{E}[\Gamma_t] + \frac{\eta^2 H}{n}(2+\frac{4r}{\lambda_2})\sum_{q=1}^{H}\sum_{i=1}^{n}\mathbb{E}\|\widetilde{h}_i^q(X_t^i)\|^2.$$

□

Lemma 12. *For any $1 \leq q \leq H$ and step T, we have that*

$$\sum_{i=1}^{n}\mathbb{E}\|\nabla f(\mu_t) - h_i^q(X_t^i)\|^2 \leq 2L^2\mathbb{E}[\Gamma_t] + \sum_{i=1}^{n}2L^2\eta^2\mathbb{E}\|\sum_{s=0}^{q-1}\widetilde{h}_i^s(X_t^i)\|^2.$$

Proof.

$$\sum_{i=1}^{n}\mathbb{E}\|\nabla f(\mu_t) - h_i^q(X_t^i)\|^2 = \sum_{i=1}^{n}\mathbb{E}\|\nabla f(\mu_t) - \nabla f(X_t^i - \sum_{s=0}^{q-1}\eta\widetilde{h}_i^s(X_t^i))\|^2$$
$$\leq \sum_{i=1}^{n}L^2\mathbb{E}\|\mu_t - X_t^i + \eta\widetilde{h}_i^{q-1}(X_t^i))\|^2 \leq \sum_{i=1}^{n}2L^2\mathbb{E}\|X_t^i - \mu_t\|^2 + \sum_{i=1}^{n}2L^2\eta^2\mathbb{E}\|\sum_{s=0}^{q-1}\widetilde{h}_i^s(X_t^i)\|^2$$
$$= 2L^2\mathbb{E}[\Gamma_t] + \sum_{i=1}^{n}2L^2\eta^2\mathbb{E}\|\sum_{s=0}^{q-1}\widetilde{h}_i^s(X_t^i)\|^2.$$

□

Lemma 13. *For any $1 \leq q \leq H$ and step T, we have that*

$$\sum_{i=1}^{n}\mathbb{E}\|\widetilde{h}_i^q(X_t^i)\|^2 \leq n\sigma^2 + 12L^2\mathbb{E}[\Gamma_t] + \sum_{i=1}^{n}12L^2\eta^2\mathbb{E}\|\sum_{s=0}^{q-1}\widetilde{h}_i^s(X_t^i)\|^2 + 3n\mathbb{E}\|\sum_{i=1}^{n}h_i^q(X_t^i)/n\|^2.$$

Proof.

$$\sum_{i=1}^{n}\mathbb{E}\|\widetilde{h}_i^q(X_t^i)\|^2 \le \sum_{i=1}^{n}(\sigma^2+\mathbb{E}\|h_i^q(X_t^i)\|^2) \le n\sigma^2+\sum_{i=1}^{n}\mathbb{E}\|\nabla f(X_t^i-\sum_{s=0}^{q-1}\eta\widetilde{h}_i^s(X_t^i))\|^2$$

$$\le n\sigma^2+\sum_{i=1}^{n}\mathbb{E}\Big\|\nabla f(X_t^i-\eta\widetilde{h}_i^{q-1}(X_t^i))-\nabla f(\mu_t)+\nabla f(\mu_t) - \sum_{j=1}^{n}\nabla f(X_t^j-\sum_{s=0}^{q-1}\eta\widetilde{h}_j^s(X_t^j))/n+\sum_{j=1}^{n}\nabla f(X_t^j-\sum_{s=0}^{q-1}\eta\widetilde{h}_j^s(X_t^j))/n\Big\|^2$$

$$\le n\sigma^2+\sum_{i=1}^{n}3\mathbb{E}\|\nabla f(\mu_t)-h_i^q(X_t^i)\|^2+3n\mathbb{E}\|\sum_{i=1}^{n}(\nabla f(\mu_t)-h_i^q(X_t^i))/n\|^2+3n\mathbb{E}\|\sum_{i=1}^{n}h_i^q(X_t^i)/n\|^2$$

$$\le n\sigma^2+\sum_{i=1}^{n}6\mathbb{E}\|\nabla f(\mu_t)-h_i^q(X_t^i)\|^2+3n\mathbb{E}\|\sum_{i=1}^{n}h_i^q(X_t^i)/n\|^2$$

$$\le n\sigma^2+12L^2\mathbb{E}[\Gamma_t]+\sum_{i=1}^{n}12L^2\eta^2\mathbb{E}\|\sum_{s=0}^{q-1}\widetilde{h}_i^s(X_t^i)\|^2+3n\mathbb{E}\|\sum_{i=1}^{n}h_i^q(X_t^i)/n\|^2.$$

□

Next we use the above lemma to show the upper bound for $\sum_{q=1}^{H}\sum_{i=1}^{n}\mathbb{E}\|\widetilde{h}_i^q(X_t^i)\|^2$:

Lemma 14. *For $\eta \le \frac{1}{6LH}$, we have that :*

$$\sum_{q=1}^{H}\sum_{i=1}^{n}\mathbb{E}\|\widetilde{h}_i^q(X_t^i)\|^2 \le 2Hn\sigma^2+24HL^2\mathbb{E}[\Gamma_t]+6n\sum_{q=1}^{H}\mathbb{E}\|\sum_{i=1}^{n}h_i^q(X_t^i)/n\|^2$$

Proof. Notice that if $\eta \le \frac{1}{6LH}$ the Lemma 13 gives us that :

$$\sum_{i=1}^{n}\mathbb{E}\|\widetilde{h}_i^q(X_t^i)\|^2 \le n\sigma^2+12L^2\mathbb{E}[\Gamma_t]+\sum_{i=1}^{n}\frac{1}{2H^2}\mathbb{E}\|\sum_{s=0}^{q-1}\widetilde{h}_i^s(X_t^i)\|^2+3n\mathbb{E}\|\sum_{i=1}^{n}h_i^q(X_t^i)/n\|^2 \tag{31}$$

$$\le n\sigma^2+12L^2\mathbb{E}[\Gamma_t]+\sum_{i=1}^{n}\frac{q}{2H^2}\sum_{s=0}^{q-1}\mathbb{E}\|\widetilde{h}_i^s(X_t^i)\|^2+3n\mathbb{E}\|\sum_{i=1}^{n}h_i^q(X_t^i)/n\|^2$$

$$\le n\sigma^2+12L^2\mathbb{E}[\Gamma_t]+\sum_{i=1}^{n}\frac{1}{2H}\sum_{s=0}^{q-1}\mathbb{E}\|\widetilde{h}_i^s(X_t^i)\|^2+3n\mathbb{E}\|\sum_{i=1}^{n}h_i^q(X_t^i)/n\|^2.$$

For $0 \le q \le H$, let

$$R_q=\sum_{i=1}^{n}\sum_{s=0}^{q}\mathbb{E}\|\widetilde{h}_i^s(X_t^i)\|^2.$$

Observe that the inequality 31 can be rewritten as:

$$R_q - R_{q-1} \leq \frac{1}{2H} R_{q-1} + n\sigma^2 + 12L^2\mathbb{E}[\Gamma_t] + 3n\mathbb{E}\|\sum_{i=1}^{n} h_i^q(X_t^i)/n\|^2.$$

which is the same as

$$R_q \leq (1 + \frac{1}{2H}) R_{q-1} + n\sigma^2 + 12L^2\mathbb{E}[\Gamma_t] + 3n\mathbb{E}\|\sum_{i=1}^{n} h_i^q(X_t^i)/n\|^2.$$

By unrolling the recursion we get that

$$R_H \leq \sum_{q=0}^{H-1} (1 + \frac{1}{2H})^q \Big(n\sigma^2 + 12L^2\mathbb{E}[\Gamma_t] + 3n\mathbb{E}\|\sum_{i=1}^{n} h_i^{H-q}(X_t^i)/n\|^2 \Big)$$

Since,

$$(1 + \frac{1}{2H})^H \leq (e^{\frac{1}{2H}})^H = e^{1/2} \leq 2$$

we have that

$$\begin{aligned} R_H &= \sum_{q=1}^{H}\sum_{i=1}^{n} \mathbb{E}\|\widetilde{h}_i^q(X_t^i)\|^2 \leq 2\Bigg(\sum_{q=1}^{H} \Big(n\sigma^2 + 12L^2\mathbb{E}[\Gamma_t] + 3n\mathbb{E}\|\sum_{i=1}^{n} h_i^q(X_t^i)/n\|^2 \Big) \Bigg) \\ &= 2Hn\sigma^2 + 24HL^2\mathbb{E}[\Gamma_t] + 6n\sum_{q=1}^{H}\mathbb{E}\|\sum_{i=1}^{n} h_i^q(X_t^i)/n\|^2. \end{aligned}$$

□

Next we derive the upper bound for $\sum_{t=0}^{T}\mathbb{E}[\Gamma_t]$:

Lemma 5. *For $\eta \leq \frac{1}{7HL\sqrt{2r/\lambda_2 + 4r^2/\lambda_2^2}}$, we have that :*

$$\sum_{t=0}^{T}\mathbb{E}[\Gamma_t] \leq \frac{4nr\eta^2\sigma^2H^2T}{\lambda_2}(2 + \frac{4r}{\lambda_2}) + \frac{12nr\eta^2H}{\lambda_2}(2 + \frac{4r}{\lambda_2})\sum_{t=1}^{T}\sum_{q=1}^{H}\mathbb{E}\|\sum_{i=1}^{n} h_i^q(X_t^i)/n\|^2.$$

Proof. By combining Lemmas 11 and 14 we get that:

$$\begin{aligned} \mathbb{E}[\Gamma_{t+1}] &\leq (1 - \frac{\lambda_2}{rn})\mathbb{E}[\Gamma_t] + \frac{\eta^2 H}{n}(2 + \frac{4r}{\lambda_2})\sum_{q=1}^{H}\sum_{i=1}^{n}\mathbb{E}\|\widetilde{h}_i^q(X_t^i)\|^2 \\ &\leq (1 - \frac{\lambda_2}{rn})\mathbb{E}[\Gamma_t] + \frac{\eta^2 H}{n}(2 + \frac{4r}{\lambda_2})\Big(2Hn\sigma^2 + 24HL^2\mathbb{E}[\Gamma_t] + 6n\sum_{q=1}^{H}\mathbb{E}\|\sum_{i=1}^{n} h_i^q(X_t^i)/n\|^2\Big) \\ &= (1 - \frac{\lambda_2}{rn})\mathbb{E}[\Gamma_t] + 2\eta^2\sigma^2H^2(2 + \frac{4r}{\lambda_2}) + \frac{24H^2L^2\eta^2}{n}(2 + \frac{4r}{\lambda_2})\mathbb{E}[\Gamma_t] + 6\eta^2H(2 + \frac{4r}{\lambda_2})\sum_{q=1}^{H}\mathbb{E}\|\sum_{i=1}^{n} h_i^q(X_t^i)/n\| \end{aligned}$$

Notice that for $\eta \leq \frac{1}{7HL\sqrt{2r/\lambda_2+4r^2/\lambda_2^2}}$ we can rewrite the above inequality as

$$\mathbb{E}[\Gamma_{t+1}] \leq (1-\frac{\lambda_2}{2nr})\mathbb{E}[\Gamma_t]+2\eta^2\sigma^2H^2(2+\frac{4r}{\lambda_2})+6\eta^2H(2+\frac{4r}{\lambda_2})\sum_{q=1}^{H}\mathbb{E}\|\sum_{i=1}^{n}h_i^q(X_t^i)/n\|^2.$$

since $\sum_{i=0}^{\infty}(1-\frac{\lambda_2}{2nr})^i \leq \frac{1}{1-(1-\frac{\lambda_2}{2nr})} = \frac{2nr}{\lambda_2}$ we get that:

$$\begin{aligned}\sum_{t=0}^{T}\mathbb{E}[\Gamma_t] &\leq \frac{2nr}{\lambda_2}\Big(2\eta^2\sigma^2H^2(2+\frac{4r}{\lambda_2})+6\eta^2H(2+\frac{4r}{\lambda_2})\sum_{q=1}^{H}\mathbb{E}\|\sum_{i=1}^{n}h_i^q(X_t^i)/n\|^2\Big)\\ &= \frac{4nr\eta^2\sigma^2H^2T}{\lambda_2}(2+\frac{4r}{\lambda_2})+\frac{12nr\eta^2H}{\lambda_2}(2+\frac{4r}{\lambda_2})\sum_{t=1}^{T}\sum_{q=1}^{H}\mathbb{E}\|\sum_{i=1}^{n}h_i^q(X_t^i)/n\|^2.\end{aligned}$$

□

Now, we are ready to prove the following theorem

Theorem 4. *Let f be an non-convex, L-smooth, function satisfying condition 4, whose minimum $x^\star$ we are trying to find via the SwarmSGD procedure given in Algorithm 1. Let H be the number of local stochastic gradient steps performed by each agent upon interaction. Define $\mu_t = \sum_{i=1}^{n}X_t^i/n$, where X_t^i is a value of model i after t interactions. For learning rate $\eta = \frac{n}{\sqrt{T}}$ and $T \geq 225n^4H^2\max(1,(2r/\lambda_2+4r^2/\lambda_2^2)^2)\max(1,L^2)$ we have that:*

$$\frac{\sum_{i=0}^{T-1}\mathbb{E}\|\nabla f(\mu_t)\|^2}{T} \leq \frac{1}{\sqrt{T}H}\mathbb{E}[f(\mu_0)-f(\mu_t)]+\frac{28H^2\max(1,L^2)\sigma^2}{\sqrt{T}}\max(1,2r/\lambda_2+4r^2/\lambda_2^2).$$

Proof.

$$\mathbb{E}[f(\mu_{t+1})] \overset{L-smoothness}{\leq} \mathbb{E}[f(\mu_t)] + \mathbb{E}\langle \nabla f(\mu_t), \mu_{t+1} - \mu_t \rangle + \frac{L}{2}\mathbb{E}\|\mu_{t+1} - \mu_t\|^2 \tag{32}$$

$$= \mathbb{E}[f(\mu_t)] + \sum_{(i,j)\in E} \frac{1}{rn/2}\mathbb{E}\langle \nabla f(\mu_t), -\frac{\eta}{n}\widetilde{h}_i(X_t^i) - \frac{\eta}{n}\widetilde{h}_j(X_t^j)\rangle \tag{33}$$

$$+ \sum_{(i,j)\in E} \frac{L}{rn}\mathbb{E}\|\frac{\eta}{n}\widetilde{h}_i(X_t^i) + \frac{\eta}{n}\widetilde{h}_j(X_t^j)\|^2 \tag{34}$$

$$\leq \mathbb{E}[f(\mu_t)] + \sum_{(i,j)\in E} \frac{1}{rn/2}\mathbb{E}\langle \nabla f(\mu_t), -\frac{\eta}{n}\widetilde{h}_i(X_t^i) - \frac{\eta}{n}\widetilde{h}_j(X_t^j)\rangle$$

$$+ \sum_{(i,j)\in E} \frac{2L}{rn}\mathbb{E}\Big[\|\frac{\eta}{n}\widetilde{h}_i(X_t^i)\|^2 + \|\frac{\eta}{n}\widetilde{h}_j(X_t^j)\|^2\Big] \tag{35}$$

$$= \mathbb{E}[f(\mu_t)] + \sum_{i=1}^{n} \frac{2}{n}\mathbb{E}\langle \nabla f(\mu_t), -\sum_{q=1}^{H}\frac{\eta}{n}h_i^q(X_t^i)\rangle + \sum_{i=1}^{n}\frac{2L}{n}\mathbb{E}\|\frac{\eta}{n}\widetilde{h}_i(X_t^i)\|^2 \tag{36}$$

$$= \mathbb{E}[f(\mu_t)] + \sum_{q=1}^{H}\sum_{i=1}^{n} \frac{2}{n}\mathbb{E}\langle \nabla f(\mu_t), -\frac{\eta}{n}h_i^q(X_t^i)\rangle + \sum_{q=1}^{H}\sum_{i=1}^{n}\frac{2LH}{n}\mathbb{E}\|\frac{\eta}{n}\widetilde{h}_i^q(X_t^i)\|^2 \tag{37}$$

$$= \mathbb{E}[f(\mu_t)] + \sum_{q=1}^{H}\Bigg(\frac{\eta}{n}\mathbb{E}\|\sum_{i=1}^{n}(\nabla f(\mu_t) - h_i^q(X_t^i))/n\|^2 - \frac{\eta}{n}\mathbb{E}\|\nabla f(\mu_t)\|^2 - \frac{\eta}{n}\mathbb{E}\|\sum_{i=1}^{n}h_i^q(X_t^i)/n\|^2$$

$$+ \sum_{i=1}^{n}\frac{2LH}{n}\mathbb{E}\|\frac{\eta}{n}\widetilde{h}_i^q(X_t^i)\|^2\Bigg) \tag{38}$$

$$\leq \mathbb{E}[f(\mu_t)] + \sum_{q=1}^{H}\Bigg(\frac{\eta}{n^2}\sum_{i=1}^{n}\mathbb{E}\|\nabla f(\mu_t) - h_i^q(X_t^i)\|^2 - \frac{\eta}{n}\mathbb{E}\|\nabla f(\mu_t)\|^2 - \frac{\eta}{n}\mathbb{E}\|\sum_{i=1}^{n}h_i^q(X_t^i)/n\|^2 \tag{39}$$

$$+ \sum_{i=1}^{n}\frac{2LH}{n}\mathbb{E}\|\frac{\eta}{n}\widetilde{h}_i^q(X_t^i)\|^2\Bigg) \tag{40}$$

$$\overset{Lemma12}{\leq} \mathbb{E}[f(\mu_t)] + \sum_{q=1}^{H}\Bigg(\frac{\eta}{n^2}\Big(2L^2\mathbb{E}[\Gamma_t] + \sum_{i=1}^{n}2L^2\eta^2\mathbb{E}\|\widetilde{h}_i^{q-1}(X_t^i)\|^2\Big) - \frac{\eta}{n}\mathbb{E}\|\nabla f(\mu_t)\|^2$$

$$- \frac{\eta}{n}\mathbb{E}\|\sum_{i=1}^{n}h_i^q(X_t^i)/n\|^2 + \sum_{i=1}^{n}\frac{2LH}{n}\mathbb{E}\|\frac{\eta}{n}\widetilde{h}_i^q(X_t^i)\|^2\Bigg) \tag{41}$$

$$\overset{Lemma14}{\leq} \mathbb{E}[f(\mu_t)] + \frac{2\eta L^2 H}{n^2}\mathbb{E}[\Gamma_t] - \frac{H\eta}{n}\mathbb{E}\|\nabla f(\mu_t)\|^2 \tag{42}$$

$$+ \frac{2L^2\eta^3}{n^2}\Big(2Hn\sigma^2 + 24HL^2\mathbb{E}[\Gamma_t] + 6n\sum_{q=1}^{H}\mathbb{E}\|\sum_{i=1}^{n}h_i^q(X_t^i)/n\|^2\Big)$$

$$+ \frac{2LH\eta^2}{n^3}\Big(2Hn\sigma^2 + 24HL^2\mathbb{E}[\Gamma_t] + 6n\sum_{q=1}^{H}\mathbb{E}\|\sum_{i=1}^{n}h_i^q(X_t^i)/n\|^2\Big)$$

$$- \frac{\eta}{n}\sum_{q=1}^{H}\mathbb{E}\|\sum_{i=1}^{n}h_i^q(X_t^i)/n\|^2. \tag{43}$$

Next we choose $\eta \leq \frac{1}{8L}$ and $\eta \leq \frac{n}{60LH}$, so that $\frac{12L^2\eta^3}{n} \leq \frac{\eta}{5n}$ and $\frac{12LH\eta^2}{n^2} \leq \frac{\eta}{5n}$. This together with the above inequalities allows us to derive the following upper bound for $\mathbb{E}[f(\mu_{t+1})]$(we eliminate terms with positive multiplicative factor $\mathbb{E}\|\sum_{i=1}^n h_i^q(X_t^i)/n\|^2$):

$$\begin{aligned}\mathbb{E}[f(\mu_{t+1})] \leq \mathbb{E}[f(\mu_t)] + \frac{2\eta L^2 H}{n^2}\mathbb{E}[\Gamma_t] - \frac{H\eta}{n}\mathbb{E}\|\nabla f(\mu_t)\|^2 \\ + \frac{2L^2\eta^3}{n^2}\Big(2Hn\sigma^2 + 24HL^2\mathbb{E}[\Gamma_t]\Big) + \frac{2LH\eta^2}{n^3}\Big(2Hn\sigma^2 + 24HL^2\mathbb{E}[\Gamma_t]\Big) \\ - \frac{3\eta}{5n}\sum_{q=1}^H \mathbb{E}\|\sum_{i=1}^n h_i^q(X_t^i)/n\|^2.\end{aligned}$$

We proceed by summing up the above inequality for $0 \leq t \leq T-1$:

$$\begin{aligned}\sum_{t=0}^{T-1}\mathbb{E}[f(\mu_{t+1})] \leq \sum_{t=0}^{T-1}\mathbb{E}[f(\mu_t)] + \frac{4L^2H\eta^3\sigma^2 T}{n} + \frac{4LH^2\eta^2\sigma^2 T}{n^2} - \sum_{i=0}^{T-1}\frac{\eta H}{n}\mathbb{E}\|\nabla f(\mu_t)\|^2 \\ + \frac{2\eta L^2 H}{n^2}\sum_{t=0}^{T-1}\mathbb{E}[\Gamma_t] + \frac{48L^4H\eta^3}{n^2}\sum_{t=0}^{T-1}\mathbb{E}[\Gamma_t] + \frac{48L^3H^2\eta^2}{n^3}\sum_{t=0}^{T-1}\mathbb{E}[\Gamma_t] \\ - \sum_{t=0}^{T-1}\frac{3\eta}{5n}\sum_{q=1}^H\mathbb{E}\|\sum_{i=1}^n h_i^q(X_t^i)/n\|^2 \qquad (44)\end{aligned}$$

Next we use Lemma 5:

$$\begin{aligned}\frac{2\eta L^2 H}{n^2}\sum_{t=0}^{T-1}\mathbb{E}[\Gamma_t] + \frac{48L^4H\eta^3}{n^2}\sum_{t=0}^{T-1}\mathbb{E}[\Gamma_t] + \frac{48L^3H^2\eta^2}{n^3}\sum_{t=0}^{T-1}\mathbb{E}[\Gamma_t] - \sum_{t=0}^{T-1}\frac{3\eta}{5n}\sum_{q=1}^H\mathbb{E}\|\sum_{i=1}^n h_i^q(X_t^i)/n\|^2 \\ \leq \frac{2\eta L^2 H}{n^2}\Big(\frac{4nr\eta^2\sigma^2H^2T}{\lambda_2}(2+\frac{4r}{\lambda_2}) + \frac{12nr\eta^2H}{\lambda_2}(2+\frac{4r}{\lambda_2})\sum_{t=1}^{T-1}\sum_{q=1}^H\mathbb{E}\|\sum_{i=1}^n h_i^q(X_t^i)/n\|^2\Big) \\ + \frac{48L^4H\eta^3}{n^2}\Big(\frac{4nr\eta^2\sigma^2H^2T}{\lambda_2}(2+\frac{4r}{\lambda_2}) + \frac{12nr\eta^2H}{\lambda_2}(2+\frac{4r}{\lambda_2})\sum_{t=1}^{T-1}\sum_{q=1}^H\mathbb{E}\|\sum_{i=1}^n h_i^q(X_t^i)/n\|^2\Big) \\ + \frac{48L^3H^2\eta^2}{n^3}\Big(\frac{4nr\eta^2\sigma^2H^2T}{\lambda_2}(2+\frac{4r}{\lambda_2}) + \frac{12nr\eta^2H}{\lambda_2}(2+\frac{4r}{\lambda_2})\sum_{t=1}^{T-1}\sum_{q=1}^H\mathbb{E}\|\sum_{i=1}^n h_i^q(X_t^i)/n\|^2\Big) \\ - \sum_{t=0}^{T-1}\frac{3\eta}{5n}\sum_{q=1}^H\mathbb{E}\|\sum_{i=1}^n h_i^q(X_t^i)/n\|^2.\end{aligned}$$

By choosing $\eta \leq \frac{1}{11HL\sqrt{2r/\lambda_2+4r^2/\lambda_2^2}}$, $\eta \leq \frac{1}{8H^{1/2}L(2r/\lambda_2+4r^2/\lambda_2^2)^{1/4}}$ and $\eta \leq \frac{n^{1/3}}{15LH(2r/\lambda_2+4r^2/\lambda_2^2)^{1/3}}$ we can eliminate terms with the multiplicative factor

$\sum_{q=1}^{H}\mathbb{E}\|\sum_{i=1}^{n}h_i^q(X_t^i)/n\|^2$ in the above inequality:

$$\frac{2\eta L^2 H}{n^2}\sum_{t=0}^{T-1}\mathbb{E}[\Gamma_t]+\frac{48L^4H\eta^3}{n^2}\sum_{t=0}^{T-1}\mathbb{E}[\Gamma_t]+\frac{48L^3H^2\eta^2}{n^3}\sum_{t=0}^{T-1}\mathbb{E}[\Gamma_t]-\sum_{t=0}^{T-1}\frac{3\eta}{5n}\sum_{q=1}^{H}\mathbb{E}\|\sum_{i=1}^{n}h_i^q(X_t^i)/n\|^2$$
$$\leq\frac{8\eta^3L^2H^3T\sigma^2}{n}(2r/\lambda_2+4r^2/\lambda_2^2)+\frac{192L^4H^3\eta^5T\sigma^2}{n}(2r/\lambda_2+4r^2/\lambda_2^2)+\frac{192L^3H^4\eta^4\sigma^2T}{n^2}(2r/\lambda_2+4r^2/$$

By using the above inequality in inequality 44, we get that :

$$\sum_{t=0}^{T-1}\mathbb{E}[f(\mu_{t+1})]\leq\sum_{t=0}^{T-1}\mathbb{E}[f(\mu_t)]-\sum_{i=0}^{T-1}\frac{\eta H}{n}\mathbb{E}\|\nabla f(\mu_t)\|^2$$
$$+\frac{4L^2H\eta^3\sigma^2T}{n}+\frac{4LH^2\eta^2\sigma^2T}{n^2}+\frac{8\eta^3L^2H^3T\sigma^2}{n}(2r/\lambda_2+4r^2/\lambda_2^2)$$
$$+\frac{192L^4H^3\eta^5T\sigma^2}{n}(2r/\lambda_2+4r^2/\lambda_2^2)+\frac{192L^3H^4\eta^4\sigma^2T}{n^2}(2r/\lambda_2+4r^2/\lambda_2^2).$$

After rearranging terms and dividing by $\frac{\eta TH}{n}$ we get that

$$\frac{\sum_{i=0}^{T-1}\mathbb{E}\|\nabla f(\mu_t)\|^2}{T}\leq\frac{n}{\eta TH}\mathbb{E}[f(\mu_0)-f(\mu_t)]+4L^2\eta^2\sigma^2+\frac{4LH\eta\sigma^2}{n}+8\eta^2L^2H^2\sigma^2(2r/\lambda_2+4r^2/\lambda_2^2)$$
$$+192L^4H^2\eta^4\sigma^2(2r/\lambda_2+4r^2/\lambda_2^2)+\frac{192L^3H^3\eta^3\sigma^2}{n}(2r/\lambda_2+4r^2/\lambda_2^2).$$

Next we use $\eta\leq 1/n$ and $\eta\leq\frac{1}{6HL}$:

$$\frac{\sum_{i=0}^{T-1}\mathbb{E}\|\nabla f(\mu_t)\|^2}{T}\leq\frac{n}{\eta TH}\mathbb{E}[f(\mu_0)-f(\mu_t)]+\frac{4L^2\eta\sigma^2}{n}+\frac{4LH\eta\sigma^2}{n}+\frac{8\eta L^2H^2\sigma^2}{n}(2r/\lambda_2+4r^2/\lambda_2^2)$$
$$+\frac{192L^4H^2\eta^3\sigma^2}{n}(2r/\lambda_2+4r^2/\lambda_2^2)+\frac{192L^3H^3\eta^3\sigma^2}{n}(2r/\lambda_2+4r^2/\lambda_2^2)$$
$$\leq\frac{n}{\eta TH}\mathbb{E}[f(\mu_0)-f(\mu_t)]+\frac{4L^2\eta\sigma^2}{n}+\frac{4LH\eta\sigma^2}{n}+\frac{8\eta L^2H^2\sigma^2}{n}(2r/\lambda_2+4r^2/\lambda_2^2)$$
$$+\frac{62L^2\eta\sigma^2}{n}(2r/\lambda_2+4r^2/\lambda_2^2)+\frac{6LH\eta\sigma^2}{n}(2r/\lambda_2+4r^2/\lambda_2^2)$$

.

Recall that $\eta=\frac{n}{\sqrt{T}}$ to get:

$$\frac{\sum_{i=0}^{T-1}\mathbb{E}\|\nabla f(\mu_t)\|^2}{T}\leq\frac{1}{\sqrt{T}H}\mathbb{E}[f(\mu_0)-f(\mu_t)]+\frac{4L^2\sigma^2}{\sqrt{T}}+\frac{4LH\sigma^2}{\sqrt{T}}+\frac{8L^2H^2\sigma^2}{\sqrt{T}}(2r/\lambda_2+4r^2/\lambda_2^2)$$
$$+\frac{6L^2\sigma^2}{\sqrt{T}}(2r/\lambda_2+4r^2/\lambda_2^2)+\frac{6LH\sigma^2}{\sqrt{T}}(2r/\lambda_2+4r^2/\lambda_2^2)$$
$$\leq\frac{1}{\sqrt{T}H}\mathbb{E}[f(\mu_0)-f(\mu_t)]+\frac{28H^2\max(1,L^2)\sigma^2}{\sqrt{T}}\max(1,2r/\lambda_2+4r^2/\lambda_2^2).$$

Notice that all assumptions and upper bounds on η are satisfied if $\eta\leq\frac{1}{15nH\max(1,(2r/\lambda_2+4r^2/\lambda_2^2))\max(1,L)}$, which is true for $T\geq 225n^4H^2\max(1,(2r/\lambda_2+4r^2/\lambda_2^2)^2)\max(1,L^2)$. □

F Splitting data among the agents

Our analysis can be extended to the case where agents do not necessarily have access to the entire data. More formally, let $f(x) = \sum_{i=1}^{n} f_i(x)/n$, where $f_i(x)$ is a loss function computed over the data available to the agent i. We will require the following assumptions(Which match the assumptions in [23]):

(1) For each agent i, the gradient $\nabla f_i(x)$ is L-Lipschitz continuous for some $L > 0$, i.e. for all $x, y \in \mathbb{R}^d$:

$$\|\nabla f_i(x) - \nabla f_i(y)\| \leq L\|x - y\|. \tag{45}$$

(2) For each agent i and $x \in \mathbb{R}^d$:

$$\mathbb{E}[\widetilde{g}_i(x)] = \nabla f_i(x). \tag{46}$$

(3) For each agent i and $x \in \mathbb{R}^d$ there exist ς^2 such that:

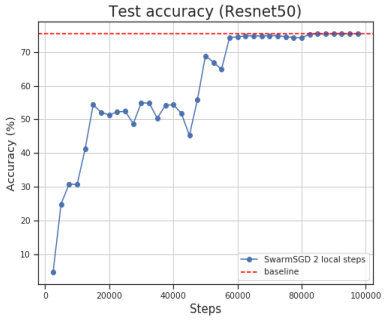
$$\mathbb{E}\|\widetilde{g}_i(x) - \nabla f_i(x)\|^2 \leq \varsigma^2. \tag{47}$$

(4) For each $x \in \mathbb{R}^d$ there exist ρ^2 such that:

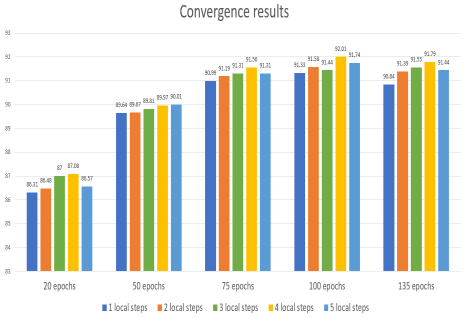
$$\sum_{i=1}^{n} \|\nabla f_i(x) - \nabla f(x)\|^2/n \leq \rho^2. \tag{48}$$

Under assumptions 45 and 46 we are able to show that Theorem 1 still holds. In the case of Theorem 4 , if all the assumptions above hold(we do not need assumption 4 in this case), we are able to derive similar result by replacing σ^2 with $O(\varsigma^2 + \rho^2)$ in the convergence bound.

G Additional Experimental Results



(a) Convergence of ResNet50/ImageNet versus Local Steps. SwarmSGD is able to recover (and slightly surpass) the Torchvision model's (baseline) top accuracy.



(b) Accuracy versus local epochs and local steps for CIFAR-10/ResNet20. The original schedule for this model has 300 epochs, and this experiment is executed on 8 nodes. If the convergence scaling were perfect, $300/8 = 37.5$ epochs would have been sufficient to converge. However, in this case we need an epoch multiplier of 2, leading to 75 epochs to recover full accuracy (which in this case is 91.5%.

Figure 2: Additional convergence results for CIFAR-10 and ImageNet datasets.