Averaging Rate Scheduler for Decentralized Learning on Heterogeneous Data

Sai Aparna Aketi, Sakshi Choudhary & Kaushik Roy
Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47906, USA.
{saketi,choudh23,kaushik}@purdue.edu

Abstract

State-of-the-art decentralized learning algorithms typically require the data distribution to be Independent and Identically Distributed (IID). However, in practical scenarios, the data distribution across the agents can have significant heterogeneity. In this work, we propose averaging rate scheduling as a simple yet effective way to reduce the impact of heterogeneity in decentralized learning. Our experiments illustrate the superiority of the proposed method ($\sim 3\%$ improvement in test accuracy) compared to the conventional approach of employing a constant averaging rate.

1 Introduction

Decentralized Learning, a subset of Federated Learning, offers many advantages over the traditional centralized approach in core aspects such as data privacy, fault tolerance, and scalability without the need for a central server. Research has shown that decentralized learning algorithms can perform comparable to centralized algorithms on benchmark datasets (Lian et al., 2017) under the assumption of IID data. Recently, there have been several efforts (Aketi et al., 2023b; Vogels et al., 2021) that make algorithmic changes to bridge the performance gap between IID and non-IID data for decentralized setups. In this work, we explore an orthogonal direction of scheduling the averaging rate to handle heterogeneous data in decentralized setups.

The averaging rate or consensus step size (Tang et al., 2019; Koloskova et al., 2019) is a hyper-parameter introduced in decentralized algorithms with communication compression to control the rate at which the model parameters across the neighbors are averaged. The value of the averaging rate depends on the compression ratio and remains constant throughout the training. The higher the compression ratio, the lower the value of the averaging rate. However, the averaging rate is usually ignored when there is no communication compression and is set to a constant value of one. We hypothesize that the averaging rate is crucial in decentralized setups with heterogeneous data. In this paper, we validate two significant claims: 1. Tuning the averaging rate is critical for decentralized learning on heterogeneous data (Figure. 1), and 2. Scheduling the averaging rate during training helps improve the performance of decentralized learning on heterogeneous data (Table. 2).

2 Background and Related Work

The main goal of decentralized learning is to learn a global model utilizing data stored locally across n agents connected via a sparse graph. Traditional decentralized algorithms (Lian et al., 2017) include three steps at every iteration - (a) Local SGD update, (b) Communication, and (c) Gossip averaging. Algorithm. 1 in the appendix describes the baseline Decentralized Stochastic Gradient Descent (DSGD). Each agent computes the gradients using local data and performs an SGD update. Then, the updated parameters are communicated to the neighbors. Finally, in the gossip averaging step, the received model parameters of the neighborhood are averaged using the mixing weights.

Several works in the literature aim to improve the performance of decentralized learning for heterogeneous data. Some of these works include tracking-based methods (Koloskova et al., 2021; Takezawa et al., 2022; Aketi et al., 2023a), cross-gradients based methods (Aketi et al., 2023b; Esfandiari et al., 2021), momentum-based methods (Lin et al., 2021), changing the mixing matrix

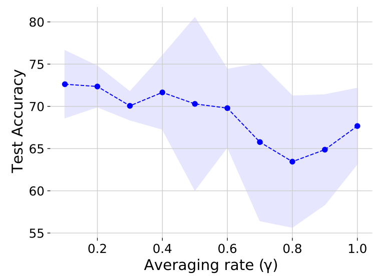


Figure 1: Variation of test accuracy with averaging rate (constant during training) during training of CIFAR-10 on ResNet-20 over a ring graph of 16 agents with $\alpha = 0.01$.

Table 1: Different Schedulers used in training of CIFAR-10 on ResNet-20 over a ring topology of 16 agents.

Scheduler	Test Accuracy
Step	70.46 ± 5.08
Exponential	$\mathbf{71.19} \pm 3.66$
Cosine	70.81 ± 5.59

Table 2: Impact of exponential Averaging Rate Scheduler (ARS) on CIFAR-10 dataset trained over a ring topology with 16 agents using DSGDm-N algorithm.

Model	w/o ARS	w/ ARS
ResNet-20	67.67 ± 4.48	$\mathbf{71.19} \pm 3.66$
VGG-11	68.38 ± 0.83	$\mathbf{71.54} \pm 1.60$

(Dandi et al., 2022) etc. However, all of these works use a constant averaging rate throughout the training and don't explore the impact of scheduling this hyper-parameter.

3 Averaging Rate Scheduler

In decentralized learning, *averaging rate* is a hyper-parameter that indicates the rate at which the model parameters are averaged across the neighborhood. Typically, the averaging rate is constant, often set to one. However, in a heterogeneous data setting, the model parameters across the neighborhood exhibit significant differences during the initial training phase. In this scenario, a high averaging rate (set to one) can disrupt local training by adding large gossip error (refer A.1 for definition) to local model parameters. To circumvent this, we propose to employ an Averaging Rate Scheduler (ARS) for decentralized learning. The proposed scheduler initializes the averaging rate to a tuned lower value and gradually increases it to one over training. Note that, unlike the learning rate, the averaging rate increases with time. Similar to the learning rate schedulers, one can explore schedulers such as exponential, step, cosine, etc. The scheduler scales the averaging rate by a factor known as *growth rate* after every k epochs. Through our exhaustive experiments, we demonstrate that the ARS improves the performance of decentralized learning on heterogeneous data.

4 Result and Conclusion

We evaluate the effectiveness of the proposed ARS on a diverse set of datasets and models over a ring topology. Following (Lin et al., 2021), we use a Dirichlet distribution-based partitioning with a parameter α quantifying the heterogeneity between the data distributions across agents. We set $\alpha = 0.01$ as it corresponds to a setting with high heterogeneity. The hyper-parameters at all the agents are synchronized at the beginning of the training (Appendix. A.2) and all the agents have the same ARS. The experiments are run for three seeds and the averaged test accuracy of the consensus model with standard deviation is presented. Figure. 1 shows that tuning the averaging rate instead of using the default value of 1 improves the performance. Note that in Figure. 1 we use a constant averaging rate throughout the training. Table. 1 shows the impact of various types of ARS on the CIFAR-10 dataset. We observe that the exponential scheduler gives the best accuracy. Table. 2 illustrates the gain in performance obtained by the addition of an exponential ARS. We observe an accuracy improvement of $\mathbf{3.5}\%$ for ResNet-20 and $\mathbf{3.2}\%$ for VGG-11 with a heterogeneous distribution of the CIFAR-10 dataset across 16 agents. The implementation details of the scheduler and the results on other datasets and models can be found in the Appendix. A.2, A.3 respectively.

In summary, this paper presents the hypothesis that the averaging rate plays a crucial role in decentralized learning on heterogeneous data, and employing an averaging rate scheduler can improve the performance of such setups. Scheduling the averaging rate (for heterogeneous data) helps to reduce the impact of neighbors' updates in the initial exploratory phase of training when the variation in model updates across the agents is high. Our experimental results corroborate the proposed hypothesis. The potential future direction is to study the impact of ARS on various decentralized learning algorithms (refer Appendix. A.4). designed for heterogeneous data.

URM Statement

The authors acknowledge that at least one key author of this work meets the URM criteria of the ICLR 2024 Tiny Papers Track.

Acknowledgements

This work was supported by the Center for the Co-Design of Cognitive Systems (COCOSYS), a DARPA-sponsored JUMP center of Semiconductor Research Corporation (SRC).

References

Sai Aparna Aketi, Abolfazl Hashemi, and Kaushik Roy. Global update tracking: A decentralized learning algorithm for heterogeneous data. In *Advances in Neural Information Processing Systems*, 2023a.

Sai Aparna Aketi, Sangamesh Kodge, and Kaushik Roy. Neighborhood gradient mean: An efficient decentralized learning method for non-iid data. *Transactions on Machine Learning Research*, 2023b. ISSN 2835-8856.

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

Yatin Dandi, Anastasia Koloskova, Martin Jaggi, and Sebastian U Stich. Data-heterogeneity-aware mixing for decentralized learning. *arXiv preprint arXiv:2204.06477*, 2022.

Yasaman Esfandiari, Sin Yong Tan, Zhanhong Jiang, Aditya Balu, Ethan Herron, Chinmay Hegde, and Soumik Sarkar. Cross-gradient aggregation for decentralized learning from non-iid data. In *International Conference on Machine Learning*, pp. 3036–3046. PMLR, 2021.

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*, pp. 770–778, 2016.

Hamel Husain. Imagenette - a subset of 10 easily classified classes from the imagenet dataset. *https://github.com/fastai/imagenette*, 2018.

Anastasia Koloskova, Tao Lin, Sebastian U Stich, and Martin Jaggi. Decentralized deep learning with arbitrary communication compression. *arXiv preprint arXiv:1907.09356*, 2019.

Anastasiia Koloskova, Tao Lin, and Sebastian U Stich. An improved analysis of gradient tracking for decentralized machine learning. *Advances in Neural Information Processing Systems*, 34: 11422–11435, 2021.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar (canadian institute for advanced research). *http://www.cs.toronto.edu/ kriz/cifar.html*, 2014.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

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. *Advances in Neural Information Processing Systems*, 30, 2017.

Tao Lin, Sai Praneeth Karimireddy, Sebastian Stich, and Martin Jaggi. Quasi-global momentum: Accelerating decentralized deep learning on heterogeneous data. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6654–6665. PMLR, 18–24 Jul 2021.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Yuki Takezawa, Han Bao, Kenta Niwa, Ryoma Sato, and Makoto Yamada. Momentum tracking: Momentum acceleration for decentralized deep learning on heterogeneous data. *arXiv preprint arXiv:2209.15505*, 2022.

Hanlin Tang, Xiangru Lian, Shuang Qiu, Lei Yuan, Ce Zhang, Tong Zhang, and Ji Liu. Deepsqueeze: Decentralization meets error-compensated compression. *arXiv preprint arXiv:1907.07346*, 2019.

Thijs Vogels, Lie He, Anastasiia Koloskova, Sai Praneeth Karimireddy, Tao Lin, Sebastian U Stich, and Martin Jaggi. Relaysum for decentralized deep learning on heterogeneous data. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 28004–28015. Curran Associates, Inc., 2021.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

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

A Appendix

A.1 Decentralized Learning

The main goal of decentralized learning is to learn a global model using the information extracted from the locally stored data across n edge agents while maintaining privacy constraints. In particular, we solve the optimization problem of minimizing global loss function $f(x)$ distributed across n agents as given in equation. 1. Here, f_i is a local loss function (for example, cross-entropy loss) computed in terms of the data sampled (d_i) from the local dataset D_i at agent i with model parameters x_i.

$$\begin{aligned} &\min_{x \in \mathbb{R}^d} f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x), \\ &and \;\; f_i(x) = \mathbb{E}_{d_i \in D_i}[F_i(x; d_i)] \;\; \forall i \end{aligned} \tag{1}$$

This is typically achieved by combining stochastic gradient descent (Bottou, 2010) with global consensus-based gossip averaging (Xiao & Boyd, 2004). The communication topology in this setup is modeled as a graph $G = ([n], E)$. An edge $\{i, j\}$ is present in E if and only if is a communication link between agents i and j to exchange the messages directly. $\mathcal{N}_i$ is used to represent the neighbors of i including itself. It is assumed that the graph G is strongly connected with self-loops. The adjacency matrix of the graph G is referred to as a mixing matrix W where w_{ij} is the weight associated with the edge $\{i, j\}$. Note that, weight 0 indicates the absence of a direct edge between the agents. We assume that the mixing matrix is doubly stochastic and symmetric, similar to all previous works in decentralized learning (Lian et al., 2017; Lin et al., 2021).

Algorithm 1 Decentralized Learning with *DSGD* (Lian et al., 2017)

Input: Each agent $i \in [1, n]$ initializes model weights $x_i^{(0)}$, step size η, averaging rate γ, and mixing matrix $W = [w_{ij}]_{i,j \in [1,n]}$, $\mathcal{N}_i$ represents neighbors of agent i including itself.

Each agent simultaneously implements the Train( ) procedure

```
1. procedure Train( )
2.     for t=$0, 1, \ldots, T-1$ do
3.         $d_i^t \sim D_i$                                                    // mini-batch sampling
4.         $g_i^t = \nabla_x F_i(d_i^t; x_i^t)$                                 // Compute local gradients
5.         $x_i^{t+\frac{1}{2}} = x_i^t - \eta g_i^t$                                     // Local SGD step
6.         Send($x_i^{t+\frac{1}{2}}$)  Receive($x_j^{t+\frac{1}{2}}$)          // Communicate parameters across neighbors
7.         $x_i^{(t+1)} = x_i^{t+\frac{1}{2}} + \gamma \sum_{j \in \mathcal{N}_i} w_{ij}(x_j^{t+\frac{1}{2}} - x_i^{t+\frac{1}{2}})$   // Gossip averaging step
8. return
```

Algorithm. 1 describes the flow of Decentralized Stochastic Gradient Descent (DSGD) which is the baseline algorithm. DSGD has three main steps at every time step - (a) Local update, (b) Communication, and (c) Gossip averaging. At every iteration, each agent computes the gradients using local data and updates its model parameters as shown in line 6 of Alg. 1. Then these updated model parameters are communicated to the neighbors as shown in line 7 of Alg. 1. Finally, in the gossip averaging step, the local model parameters are averaged with the received model parameters of the neighbors using the mixing weights (shown in line 8 of Alg. 1).

Gossip error: The term $\sum_{j \in \mathcal{N}_i} w_{ij}(x_j^{t+\frac{1}{2}} - x_i^{t+\frac{1}{2}})$ in line 7 of Alg. 1 is referred to as the *gossip error* for agent i. The gossip error is scaled by the averaging rate before adding to the local model parameters.

A.2 Experimental Details

For the decentralized setup, we use an undirected ring topology with a uniform mixing matrix. The undirected ring topology for any graph size has 3 peers per agent including itself and each edge has a mixing weight of $\frac{1}{3}$. We use the Dirichlet distribution (Lin et al., 2021) to generate heterogeneous data distribution (skewed label partition) across the agents. The created data partition across the agents is fixed, non-overlapping, and never shuffled across agents during the training. All the experiments have the degree of heterogeneity set to a value of $\alpha = 0.01$. DSGDm-N indicates the Decentralized Parallel Stochastic Gradient Descent algorithm (Lian et al., 2017) with Nesterov momentum. All the experiments presented in the paper utilize D-PSGD as the decentralized learning algorithm. The PyTorch code is available at `https://github.com/aparna-aketi/ARS`.

A.2.1 Datasets

We use the following datasets for the image classification task.

CIFAR-10: CIFAR-10 (Krizhevsky et al., 2014) is an image classification dataset with 10 classes. The image samples are colored (3 input channels) and have a resolution of 32×32. There are $50,000$ training samples with 5000 samples per class and $10,000$ test samples with 1000 samples per class.

CIFAR-100: CIFAR-100 (Krizhevsky et al., 2014) is an image classification dataset with 100 classes. The image samples are colored (3 input channels) and have a resolution of 32×32. There are $50,000$ training samples with 500 samples per class and $10,000$ test samples with 100 samples per class.

Fashion MNIST: Fashion MNIST (Xiao et al., 2017) is an image classification dataset with 10 classes. The image samples are in greyscale (1 input channel) and have a resolution of 28×28. There are $60,000$ training samples with 6000 samples per class and $10,000$ test samples with 1000 samples per class.

Imagenette: Imagenette (Husain, 2018) is a 10-class subset of the ImageNet dataset. The image samples are colored (3 input channels) and have a resolution of 224×224. There are 9469 training samples with roughly 950 samples per class and 3925 test samples.

A.2.2 Model architectures

We replace ReLU+BatchNorm layers of all the model architectures with EvoNorm-S0 as it was shown to be better suited for decentralized learning over non-IID data (Lin et al., 2021). The details of the model architectures are given below.

ResNet-20: For ResNet-20 (He et al., 2016), we use the standard architecture with $0.27M$ trainable parameters except that BatchNorm+ReLU layers are replaced by EvoNorm-S0.

VGG-11: We modify the standard VGG-11 (Simonyan & Zisserman, 2014) architecture by reducing the number of filters in each convolutional layer by $4\times$ and using only one dense layer with 128 units. Each convolutional layer is followed by EvoNorm-S0 as the activation-normalization layer. VGG-11 has $0.58M$ trainable parameters.

LeNet-5: For LeNet-5 (LeCun et al., 1998), we use the standard architecture with $61,706$ trainable parameters.

MobileNet-V2: We use the the standard MobileNet-V2 (Sandler et al., 2018) architecture used for CIFAR dataset with $2.3M$ parameters except that BatchNorm+ReLU layers are replaced by EvoNorm-S0.

A.2.3 Hyper-Parameters

All the experiments for CIFAR-10 are run for 300 epochs whereas the experiments for other datasets are run for 100 epochs. The initial learning rate is set to 0.1 for CIFAR datasets and 0.01 for Fashion MNIST and ImageNette. The learning rate is decayed by 10 at epochs 150 and 180 for CIFAR-10 and at 50 and 75 for the remaining datasets. We use Nesterov momentum with a momentum coefficient of 0.9, weight decay of 0.0001, and a mini-batch size of 32 per agent. The experiments that do not employ ARS have the averaging rate set to a constant value of 1. The results are averaged over three seeds and the average test accuracy and standard deviation of the consensus model are presented. The consensus model is obtained by averaging the model parameters of all the agents at the end of the training using the all-reduce method.

A.2.4 Averaging Rate Scheduler Details

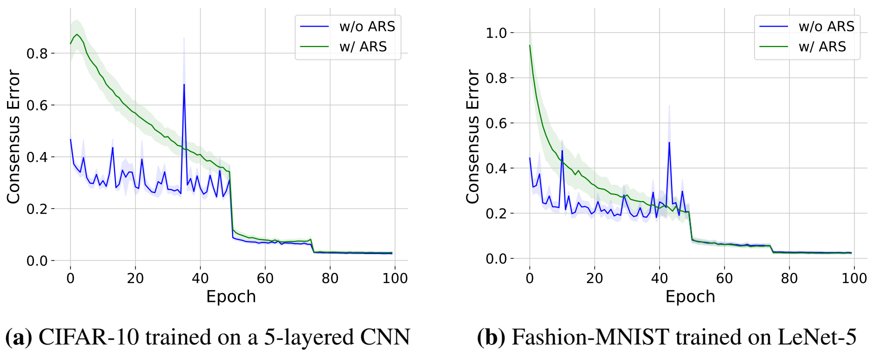
For Table. 1, we use an initial averaging rate of 0.08, and the averaging rate is gradually increased to 1 during training. For the exponential scheduler, we use a growth rate of 1.01, and k is set to 1 which means that the averaging rate is multiplied by 1.01 after every epoch. For the step scheduler, we use a growth rate of 1.09, and k is set to 10 which means that the averaging rate is multiplied by 1.09 after every 10 epochs. For both of these schedulers, we clip the averaging rate at 1 ensuring that it never takes a value greater than 1. The cosine scheduler utilizes Equation. 2. Here, we use γ_0 as 0.08 and T_{max} is 300, and k is set to 1 implying that the averaging rate is updated according to equation. 2 after every epoch.

$$\gamma_t = \gamma_0 + \frac{1}{2}(1-\gamma_0)(1-cos(\frac{t}{T_{max}}\pi)) \tag{2}$$

For Table. 2, 3, we use an exponential averaging rate for both ResNet-10 and VGG-11 models. In the case of ResNet-20 experiments, the initial averaging rate is set to 0.08 and the growth rate is set to 1.01. In the case of VGG-11, the initial averaging rate is set to 0.09 and the growth rate is set to 1.01. The hyper-parameter pair values (initial averaging rate, growth rate) for Table. 4 are Fashion-MNIST: (0.1, 1.03), CIFAR-100: (0.5, 1.02), and ImageNette: (0.1, 1.03).

A.3 Additional Results

We provide additional results on various graph sizes and datasets to show the generalizability and scalability of the ARS. Table. 3 shows the results on varying graph (ring) sizes from 16 to 48 agents for training CIFAR-10 dataset on ResNet-20 architecture. We observe that D-PSGD with ARS achieves better performance with $3.5-6.9\%$ improvement in test accuracy compared to D-PSGD without ARS.

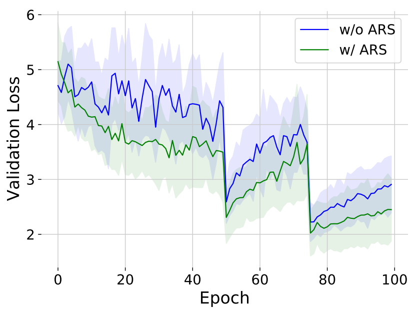


(a) CIFAR-10 trained on a 5-layered CNN **(b)** Fashion-MNIST trained on LeNet-5

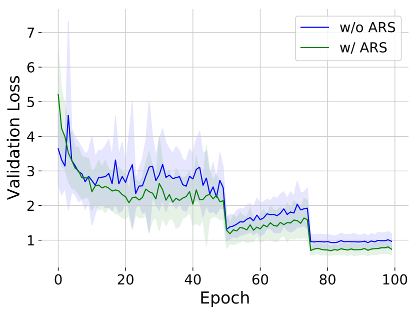
Figure 2: Average consensus error during training for various datasets trained over ring topology of 16 agents. The graph is plotted for one seed where the solid line represents the average consensus error across the agents and the shaded region represents the variation of the consensus error across agents.

Table. 4 presents the results of various datasets trained in a decentralized manner using the D-PSGD algorithm with and without ARS. We train the Fashion MNIST dataset on LeNet-5 architecture, CIFAR-100 on ResNet-20, and ImageNette on MobileNet-V2. We observe an average improvement of 2% in test accuracy across various datasets.

Additionally, we compute the average consensus error i.e., $\frac{1}{n}\sum_i^n ||x_i^t - \bar{x}||_F^2$ over time for various datasets trained over ring topology of 16 agents. $\bar{x}$ is the average/consensus model parameters. We observe that at the end of the training, the consensus error with or without ARS is similar (refer Fig. 2). The main difference is during the initial epochs of the training, where the consensus error for D-PSGD with ARS gradually decreases over time in contrast to D-PSGD without ARS which oscillates at a lower value. Note that the neighborhood's model information is not completely reliable at the initial phase of training as the models are just starting to learn and the variation in



(a) CIFAR-10 trained on a 5-layered CNN



(b) Fashion-MNIST trained on LeNet-5

Figure 3: Average validation during training for various datasets trained over ring topology of 16 agents. The graph is plotted for one seed where the solid line represents the average loss across the agents and the shaded region represents the variation of the loss across agents.

Table 3: Test accuracy of CIFAR-10 dataset trained on ResNet-20 distributed over a ring topology with a skew of $\alpha = 0.01$.

Agents	w/o ARS	w/ ARS
16	67.67 ± 4.48	$\mathbf{71.19} \pm 3.66$
24	65.98 ± 3.64	$\mathbf{69.91} \pm 3.45$
32	61.65 ± 3.51	$\mathbf{68.58} \pm 1.77$
48	56.56 ± 11.52	$\mathbf{61.79} \pm 6.28$

Table 4: Test accuracy of various datasets trained over a ring topology of 16 agents with a skew of $\alpha = 0.01$.

Dataset (Model)	w/o ARS	w/ ARS
Fashion MNIST (LeNet-5)	84.58 ± 0.32	$\mathbf{85.55} \pm 0.81$
CIFAR-100 (ResNet-20)	41.57 ± 0.72	$\mathbf{44.05} \pm 1.70$
ImageNette (MobileNet-V2)	43.68 ± 8.16	$\mathbf{46.59} \pm 4.73$

parameters across agents is high because of data heterogeneity. Since ARS starts with a lower averaging rate and exponentially increases it while training, it avoids aggressive averaging with the neighborhood's models at the initial epochs. This in turn helps in improving the performance (reducing the validation loss) of decentralized learning on heterogeneous data as shown in Fig. 3.

A.4 Limitations and Future Work

The inherent limitation of the proposed method is the hyper-parameter tuning of the initial averaging rate, growth rate, type of decay, etc. Similar to the learning rate scheduler, one needs to curate the ARS for the given dataset, model architecture, graph structure, and data distribution.

There are two potential future directions for this work:

1. A study on designing and evaluating the impacts of ARS on algorithms such as QGM (Lin et al., 2021), GUT (Aketi et al., 2023a), and (NGM_{mv}) (Aketi et al., 2023b) is a potential extension for this work. Quasi Global Momentum (QGM), Global Update Tracking (GUT), and Neighborhood Gradient Mean (NGM_{mv}) are well-known and current state-of-the-art methods for decentralized learning with heterogeneous data that do not require any communication overhead. These methods use additional memory and compute resources to improve the test accuracy and perform much better than D-PSGD and its ARS variant. ARS is a simple hyper-parameter scheduling method and can not independently compete with methods that modify the gradient or momentum information to improve performance. However, all these algorithms (QGM, GUT, and NGM_{mv}) also utilize the averaging rate hyper-parameter. Therefore, ARS can be employed in synergy with these methods to further improve the performance of decentralized learning on heterogeneous data.

2. Theoretical analysis of the convergence rate of the D-PSGD algorithm with an Averaging Rate Scheduler (ARS) can help in better understanding the impact of the ARS. Analyzing the convergence rate with ARS can shed light on whether ARS leads to faster convergence to the same solution or allows D-PSGD to converge to a better stationary point of the global objective. Note that the existing

theoretical analysis of D-PSGD (Lian et al., 2017) assumes a constant averaging rate, leaving room for potential future work.