Efficient Decentralized Deep Learning by Dynamic Model Averaging*

Michael Kamp[1,2,3**], Linara Adilova[1,2**], Joachim Sicking[1,2**], Fabian Hüger[4], Peter Schlicht[4], Tim Wirtz[1,2], and Stefan Wrobel[1,2,3]

[1] Fraunhofer IAIS <name>.<surname>@iais.fraunhofer.de
[2] Fraunhofer Center for Machine Learning
[3] University of Bonn <surname>@cs.uni-bonn.de
[4] Volkswagen Group Research <name>.<surname>@volkswagen.de

Abstract. We propose an efficient protocol for decentralized training of deep neural networks from distributed data sources. The proposed protocol allows to handle different phases of model training equally well and to quickly adapt to concept drifts. This leads to a reduction of communication by an order of magnitude compared to periodically communicating state-of-the-art approaches. Moreover, we derive a communication bound that scales well with the hardness of the serialized learning problem. The reduction in communication comes at almost no cost, as the predictive performance remains virtually unchanged. Indeed, the proposed protocol retains loss bounds of periodically averaging schemes. An extensive empirical evaluation validates major improvement of the trade-off between model performance and communication which could be beneficial for numerous decentralized learning applications, such as autonomous driving, or voice recognition and image classification on mobile phones.

1 Introduction

Traditionally, deep learning models are trained on a single system or cluster by centralizing data from distributed sources. In many applications, this requires a prohibitive amount of communication. For gradient-based training methods, communication can be reduced by calculating gradients locally and communicating the sum of gradients periodically [8], instead of raw data. This mini-batch approach performs well on tightly connected distributed systems [7, 43, 5] (e.g., data centers and clusters). For many applications, however, centralization or even periodic sharing of gradients between local devices becomes infeasible due to the large amount of necessary communication.

For decentralized systems with limited communication infrastructure it was suggested to compute local updates [45] and average models periodically, instead of sharing gradients. Averaging models has three major advantages: (i) sending

* This is a pre-print of an article submitted to ECML PKDD 2018.
** These authors contributed equally.

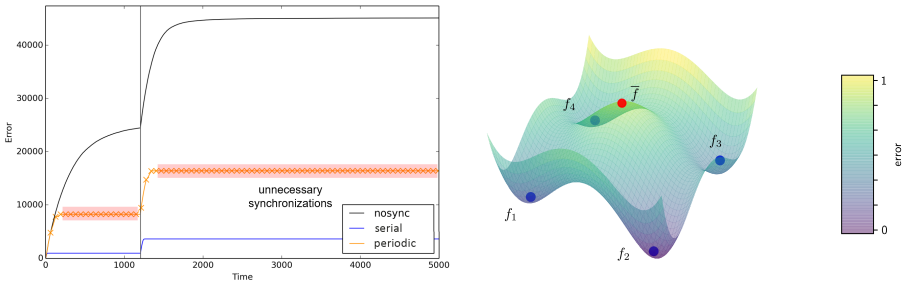


Fig. 1.1. (a) Cumulative error over time for a serial learning algorithm and two decentralized learning algorithms with 10 learners, one that does not communicate (nosync) and one that communicates every 50 time steps (periodic). The vertical line indicates a concept drift, i.e., a rapid change in the target distribution. (b) Illustration of the problem of averaging models in non-convex problems: each of the models $f_1, \ldots, f_4$ has reached a local minimum, but their average $\overline{f}$ has a higher error than each of them.

only the model parameters instead of a set of data samples reduces communication[5]; (ii) it allows to train a joint model without exchanging or centralizing privacy-sensitive data; and (iii) it can be applied to a wide range of learning algorithms, since it treats the underlying algorithm as a black-box.

This approach is used in convex optimization [33, 24, 44]. For non-convex objectives, a particular problem is that the average of a set of models can have a worse performance than any model in the set—see Figure 1.1(b). For the particular case of of deep learning, McMahan et al. [25] empirically evaluated model averaging in decentralized systems and termed it **Federated Learning**.

However, averaging periodically still invests communication independent of its utility, e.g., when all models already converged to an optimum. This disadvantage is even more apparent in case of concept drifts: periodic approaches cannot react adequately to drifts, since they either communicate so rarely that the models adapt too slowly to the change, or they communicate so frequently that they generate an immense amount of unnecessary communication in-between drifts.

In Kamp et al. [14] the authors proposed to average models dynamically, depending on the utility of the communication. The main idea is to reduce communication without losing predictive performance by investing the communication efficiently: When local learners do not suffer loss, communication is unnecessary and should be avoided (see Figure 1.1(a)); similarly, when they suffer large losses, an increased amount of communication should be invested to improve their performances. The problem setting and a criterion for efficient approaches is defined in Section 2. This approach, denoted **dynamic averaging**, was proposed for online learning convex objectives [14, 17]. We adapt dynamic averaging to the non-convex objectives of deep learning in Section 3.

[5] Note that averaging models requires the same amount of communication as sharing gradients, since the vector of model parameters is of the same dimension as the gradient vector of the loss function.

Our contribution is the description and evaluation of a general method for decentralized training of deep neural networks that (i) substantially reduces communication while retaining high predictive performance and (ii) is in addition well-suited to concept drifts in the data. To that end, Section 4 shows that, for common learning algorithms, dynamic averaging is an efficient approach for non-convex problems, i.e., it retains the predictive performance of a centralized learner but is also adaptive to the current hardness of the learning problem.

A natural application for dynamic decentralized machine learning is **in-fleet learning** of autonomous driving functionalities: concept drifts occur naturally, since properties central for the modeling task may change—changing traffic behavior both over time and different countries or regions introduce constant and unforeseeable concept drifts. Moreover, large high-frequency data streams generated by multiple sensors per vehicle renders data centralization prohibitive in large fleets. Section 5 provides an extensive empirical evaluation of the dynamic averaging approach on classical deep learning tasks, as well as synthetic and real-world tasks with concept drift, including in-fleet learning of autonomous driving functionalities. The approach is compared to periodically communicating schemes, including **Federated Averaging** [25], a state-of-the-art approach for decentralized deep learning—more recent approaches are interesting from a theoretical perspective but show no practical improvement [13], or tackle other aspects of federated learning, such as non-iid data [37] or privacy aspects [26].

Section 6 discusses properties and limitations of dynamic averaging and puts it into context of related work, followed by a conclusion in Section 7.

2 Preliminaries

We consider a decentralized learning setting with $m \in \mathbb{N}$ **local learners**, where each learner $i \in [m]$ runs the same **learning algorithm** $\varphi \colon \mathcal{F} \times 2^X \times 2^Y \to \mathcal{F}$ that trains a **local model** f^i from a **model space** $\mathcal{F}$ using local samples from an **input space** X and **output space** Y. We assume a streaming setting, where in each round $t \in \mathbb{N}$ each learner $i \in [m]$ observes a sample $E_t^i \subset X \times Y$ of size $|E_t^i| = B$, drawn iid from the same time variant distribution $P_t \colon X \times Y \to \mathbb{R}_+$. The local learner uses its local model to make a prediction whose quality is measured by a **loss function** $\ell \colon \mathcal{F} \times X \times Y \to \mathbb{R}_+$. We abbreviate the loss of the local model of learner i in round t by $\ell_t^i\left(f_t^i\right) = \sum_{(x,y)\in E_t^i} \ell\left(f_t^i, x, y\right)$[6]. The goal of decentralized learning is to minimize the **cumulative loss** up to a time horizon $T \in \mathbb{N}$, i.e.,

$$L(T,m) = \sum_{t=1}^{T} \sum_{i=1}^{m} \ell_t^i\left(f_t^i\right) \quad . \tag{1}$$

[6] This setup includes online learning ($B = 1$) and mini-batch training $B > 1$. The gradient of ℓ_t^i is the sum of individual gradients. Our approach and analysis also apply to heterogeneous sampling rates B^i for each learner i.

Guarantees on the predictive performance, measured by the cumulative loss, are typically given by a **loss bound** $\mathbf{L}(T, m)$. That is, for all possible sequences of losses it holds that $L(T, m) \leq \mathbf{L}(T, m)$.

In each round $t \in \mathbb{N}$, local learners use a **synchronization operator** $\sigma : \mathcal{F}^m \to \mathcal{F}^m$ that transfers the current set of local models, called the current **model configuration** $\mathbf{f}_t = \{f_t^1, \ldots, f_t^m\}$, into a single stronger **global model** $\sigma(\mathbf{f})$ which replaces the local models. We measure its performance in terms of communication by the cumulative communication, i.e.,

$$C(T, m) = \sum_{t=1}^{T} c(\mathbf{f}_t) \ ,$$

where $c : \mathcal{H}^m \to \mathbb{N}$ measures the number of bytes required by the protocol to synchronize the models $\mathbf{f}_t$ at time t. We investigate synchronization operators that aggregate models by computing their average [25, 24, 33, 45, 44], i.e., $\overline{f} = {}^1\!/_m \sum_{i=1}^m f^i$. In the case of neural networks, we assume that all local models have the same architecture, thus their average is the average of their respective weights. We discuss the potential use of other aggregation operations in Section 6. We denote the choice of learning algorithm together with the synchronization operator as a **decentralized learning protocol** $\Pi = (\varphi, \sigma)$. The protocol is evaluated in terms of its predictive performance and the cumulative communication it requires. In order to assess the efficiency of decentralized learning protocols in terms of the trade-off between loss and communication, Kamp et al. [16] introduced two criteria: consistency and adaptiveness.

Definition 1 (Kamp et al. [16]) *A distributed online learning protocol $\Pi = (\varphi, \sigma)$ processing mT inputs is **consistent** if it retains the loss bound of the serial online learning algorithm φ, i.e.,*

$$\boldsymbol{L}_{\Pi}(T, m) \in \mathcal{O}\left(\boldsymbol{L}_{\varphi}(mT)\right) \ .$$

*The protocol is **adaptive** if its communication bound is linear in the number of local learners m and the loss bound $\boldsymbol{L}_{\varphi}(mT)$ of the serial online learning algorithm, i.e.,*

$$C_{\Pi}(T, m) \in \mathcal{O}\left(m\boldsymbol{L}_{\varphi}(mT)\right) \ .$$

A decentralized learning protocol is **efficient** if it is both consistent and adaptive. Each one of the criteria can be trivially achieved: A non-synchronizing protocol is adaptive but not consistent, a protocol that centralizes all data is consistent but not adaptive. Protocols that communicate periodically are consistent [8, 45], i.e., they achieve a predictive performance comparable to a model that is learned centrally on all the data. However, they require an amount of communication linear in the number of learners m and the number of rounds T, independent of the loss. Thus they are not adaptive.

In the following section, we recapitulate dynamic averaging and apply it to the non-convex problem of training deep neural networks. In Section 4 we discuss in which settings it is efficient as in Definition 1.

3 Dynamic Averaging

In this section, we recapitulate the dynamic averaging protocol [17] for synchronizations based on quantifying their effect (Algorithm 1). Intuitively, communication is not well-invested in situations where all models are already approximately equal—either because they were updated to similar models or have merely changed at all since the last averaging step—and it is more effective if models are diverse. A simple measure to quantify the effect of synchronizations is given by the **divergence** of the current model configuration, i.e.,

$$\delta(\mathbf{f}) = \frac{1}{m}\sum_{i=1}^{m} \left\| f^i - \overline{f} \right\|^2 \quad . \tag{2}$$

Using this, we define the dynamic averaging operator that allows to omit synchronization in cases where the divergence of a model configuration is low.

Definition 2 (Kamp et al. [14]) *A **dynamic averaging operator** with positive divergence threshold $\Delta \in \mathbb{R}_+$ and batch size $b \in \mathbb{N}$ is a synchronization operator σ_Δ such that $\sigma_\Delta(\mathbf{f}_t) = \mathbf{f}_t$ if $t \mod b \neq 0$ and otherwise: (i) $\overline{f}_t = \overline{\sigma_\Delta(\mathbf{f}_t)}$, i.e., it leaves the mean model invariant, and (ii) $\delta\left(\sigma_\Delta(\mathbf{f})\right) \leq \Delta$, i.e., after its application the model divergence is bounded by Δ.*

An operator adhering to this definition does not generally put all nodes into sync (albeit we still refer to it as *synchronization* operator). In particular it allows to leave all models untouched as long as the divergence remains below Δ or to only average a subset of models in order to satisfy the divergence constraint.

The **dynamic averaging protocol** $\mathcal{D} = (\varphi, \sigma_\Delta)$ synchronizes the local learners using the dynamic averaging operator σ_Δ. This operator only communicates when the model divergence exceeds a **divergence threshold** Δ. In order to decide when to communicate locally, at round $t \in \mathbb{N}$, each local learner $i \in [m]$ monitors the **local condition** $\|f_t^i - r\|^2 \leq \Delta$ for a **reference model** $r \in \mathcal{H}$[36] that is common among all learners (see [19, 35, 11, 22, 18] for a more general description of this method). The local conditions guarantee that if none of them is violated, i.e., for all $i \in [m]$ it holds that $\|f_t^i - r\|^2 \leq \Delta$, then the divergence does not exceed the threshold, i.e., $\delta(\mathbf{f}_t) \leq \Delta$ [14, Theorem 6]. The closer the reference model is to the true average of local models, the tighter are the local conditions. Thus, the first choice for the reference model is the average model from the last synchronization step. The local condition is checked every $b \in \mathbb{N}$ rounds. This allows using the common mini-batch approach [3] for training deep neural networks.

If one or more local conditions are violated, all local models can be averaged—an operation referred to as full synchronization. However, on a local violation the divergence threshold is not necessarily crossed. In that case, the violations may be locally balanced: the coordinator incrementally queries other local learners for their models; if the average of all received models lies within the safe zone, it is transferred back as new model to all participating nodes. If all nodes have been queried, the result is equal to a full synchronization and the reference vector is

Algorithm 1: Dynamic Averaging Protocol

Input: divergence threshold Δ, batch size b

Initialization:
local models $f_1^1, \ldots, f_1^m \leftarrow$ one random f
reference vector $r \leftarrow f$
violation counter $v \leftarrow 0$

Round t **at node** i**:**
observe $E_t^i \subset X \times Y$
update f_{t-1}^i using the learning algorithm φ
if $t \mod b = 0$ **and** $\|f_t^i - r\|^2 > \Delta$ **then**
send f_t^i to coordinator (violation)

At coordinator on violation:
let $\mathcal{B}$ be the set of nodes with violation
$v \leftarrow v + |\mathcal{B}|$
if $v = m$ **then** $\mathcal{B} \leftarrow [m]$, $v \leftarrow 0$
while $\mathcal{B} \neq [m]$ **and** $\left\| \frac{1}{\mathcal{B}} \sum_{i \in \mathcal{B}} f_t^i - r \right\|^2 > \Delta$ **do**
augment $\mathcal{B}$ by augmentation strategy
receive models from nodes added to $\mathcal{B}$
send model $\overline{f} = \frac{1}{\mathcal{B}} \sum_{i \in \mathcal{B}} f_t^i$ to nodes in $\mathcal{B}$
if $\mathcal{B} = [m]$ also set new reference vector $r \leftarrow \overline{f}$

updated. In both cases, the divergence of the model configuration is bounded by Δ at the end of the balancing process, because all local conditions hold. Also, it is easy to check that this protocol leaves the global mean model unchanged. Hence, it is complying to Def. 2. In the following Section, we theoretically analyze the loss and communication of dynamic averaging.

4 Efficiency of Dynamic Averaging

In order to assess the predictive performance and communication cost of the dynamic averaging protocol for deep learning, we compare it to a periodically averaging approach: Given an online learning algorithm φ, the **periodic averaging protocol** $\mathcal{P} = (\varphi, \sigma_b)$ synchronizes the current model configuration $\mathbf{f}$ every $b \in \mathbb{N}$ time steps by replacing all local models by their joint average $\overline{f} = {}^1\!/_m \sum_{i=1}^m f^i$. That is, the synchronization operator is given by

$$\sigma_b(\mathbf{f}_t) = \begin{cases} \left(\overline{f}_t, \ldots, \overline{f}_t\right), & \text{if } b \equiv O(t) \\ \mathbf{f}_t = (f_t^1, \ldots, f_t^m), & \text{otherwise} \end{cases} .$$

A special case of this is the **continuous averaging protocol** $\mathcal{C} = (\varphi, \sigma_1)$, synchronizing every round, i.e., $\sigma_1(\mathbf{f}) = \left(\overline{f}, \ldots, \overline{f}\right)$. As base learning algorithm we use mini-batch SGD algorithm $\varphi_{B,\eta}^{\text{mSGD}}$ [8] commonly used in deep learning [3].

One step of this learning algorithm given the model $f \in \mathcal{F}$ can be expressed as

$$\varphi_{B,\eta}^{\text{mSGD}}(f) = f - \eta \sum_{j=1}^{B} \nabla \ell^j(f) \ .$$

Let $\mathcal{C}^{\text{mSGD}} = (\varphi_{B,\eta}^{\text{mSGD}}, \sigma_1)$ denote continuous averaging using mini-batch SGD with mini-batch size $B \in \mathbb{N}$ and learning rate $\eta \in \mathbb{R}_+$. For $m \in \mathbb{N}$ learners with the same model $f \in \mathcal{F}$, mB training samples $(x_1, y_1), \ldots, (x_{mB}, y_{mB})$, and corresponding loss functions $\ell^i(\cdot) = \ell(\cdot, x_i, y_i)$, one step of $\mathcal{C}^{\text{mSGD}}$ is

$$\sigma_1\left(\left(\varphi_{B,\eta}^{\text{mSGD}}(f), \ldots, \varphi_{B,\eta}^{\text{mSGD}}(f)\right)\right) = \frac{1}{m} \sum_{i=1}^{m} \left(f - \eta \textstyle\sum_{j=1}^{B} \nabla \ell^{(i-1)B+j}(f)\right) \ .$$

We compare $\mathcal{C}^{\text{mSGD}}$ to the serial application of mini-batch SGD. It can be observed that continuous averaging with mini-batch SGD on $m \in \mathbb{N}$ learners with mini-batch size B is equivalent to serial mini-batch SGD with a mini-batch size of mB and a learning rate that is m times smaller.

Proposition 3 *For $m \in \mathbb{N}$ learners, a mini-batch size $B \in \mathbb{N}$, mB training samples $(x_1, y_1), \ldots, (x_{mB}, y_{mB})$, corresponding loss functions $\ell^i(\cdot) = \ell(\cdot, x_i, y_i)$, a learning rate $\eta \in \mathbb{R}_+$, and a model $f \in \mathcal{F}$, it holds that*

$$\sigma_1\left(\left(\varphi_{B,\eta}^{mSGD}(f), \ldots, \varphi_{B,\eta}^{mSGD}(f)\right)\right) = \varphi_{mB,\eta/m}^{mSGD}(f) \ .$$

Proof.

$$\begin{aligned}
\sigma_1\left(\left(\varphi_{B,\eta}^{\text{mSGD}}(f), \ldots, \varphi_{B,\eta}^{\text{mSGD}}(f)\right)\right) &= \frac{1}{m} \sum_{i=1}^{m} \left(f - \eta \sum_{j=1}^{B} \nabla \ell^{(i-1)B+j}(f)\right) \\
&= \frac{1}{m} mf - \frac{1}{m} \eta \sum_{i=1}^{m} \sum_{j=1}^{B} \nabla \ell^{(i-1)B+j}(f) = f - \frac{1}{m} \eta \sum_{j=1}^{mB} \nabla \ell^j(f) = \varphi_{mB,\eta/m}^{\text{mSGD}}(f)
\end{aligned}$$

□

In particular, Proposition 3 holds for continuous averaging with a mini-batch size of $B = 1$, i.e., classic stochastic gradient descent. From Proposition 3 it follows that continuous averaging is consistent as in Definition 1, since it retains the loss bound of serial mini-batch SGD and classic SGD. If the loss function is locally convex in an $\mathcal{O}(\Delta)$-radius around the current average—a non-trivial but realistic assumption [29, 20]—Theorem 2 in Boley et al. [2] guarantees that for SGD, dynamic averaging has a predictive performance similar to any periodically communicating protocol, in particular to σ_1 (see Appendix B for details). For this case it follows that dynamic averaging using SGD for training deep neural networks is consistent. Theorem 2 in Kamp et al. [16] shows that the

communication of the dynamic averaging protocol using SGD and a divergence threshold Δ is bounded by

$$C(T,m) \in \mathcal{O}\left(\frac{c(\mathbf{f})}{\sqrt{\Delta}} L(T,m)\right) \quad ,$$

where $c(\mathbf{f})$ is the number of bytes required to be communicated to average a set of deep neural networks. Since each neural network has a fixed number of weights, this cost is in $\mathcal{O}(m)$. It follows that dynamic averaging is adaptive. Thus, using dynamic averaging with stochastic gradient descent for the decentralized training of deep neural networks is efficient as in Definition 1.

Note that the synchronization operator can be implemented using different assumptions on the system's topology and communication protocol, i.e., in a peer-to-peer fashion, or in a hierarchical communication scheme. For simplicity, in our analysis of the communication of different synchronization operators we assume that the synchronization operation is performed by a dedicated coordinator node. This coordinator is able to poll local models, aggregate them and send the global model to the local learners.

5 Empirical Evaluation

This section empirically evaluates dynamic averaging for training deep neural networks. The theoretical analysis in Section 4 indicates that dynamic averaging can achieve similar predictive performance with substantially less communication. To emphasize these theoretical arguments, we study the performance of the dynamic averaging protocol for a non-convex training objective, followed by a comparison of our approach with a state-of-the-art communication approach[7].

The performance is then evaluated in the presence of concept drifts. Combining the aforementioned aspects, we apply our protocol to a non-convex objective with possible concept drifts from the field of autonomous driving.

Throughout this section, if not specified separately, we consider mini-batch SGD φ^{mSGD} as learning algorithm, since recent studies indicate that it is particularly suited for learning deep neural networks [42]. That is, we consider communication protocols $\Pi = (\varphi^{\text{mSGD}}_{B,\eta}, \sigma)$ with various synchronization operators σ. The hyper-parameters of the protocols and the mini-batch SGD have been optimized on an independent dataset. Details on the experiments, including network architectures, can be found in the Appendix A.

Dynamic Averaging for Training Deep Neural Networks: To evaluate the performance of dynamic averaging in deep learning, we first compare it to periodic averaging for training a convolutional neural network (CNN) on the MNIST classification dataset [23]. We furthermore compare both protocols to a non-synchronizing protocol, denoted **nosync**, and a serial application of the learning algorithm on all data, denoted **serial**.

[7] The code of the experiments is available at https://bitbucket.org/Michael_Kamp/decentralized-machine-learning.

Figure 5.1 shows the cumulative error of several setups of dynamic and periodic averaging, as well as the nosync and serial baselines. The experiment confirms that for each setup of the periodic averaging protocol a setup of dynamic averaging can be found that reaches a similar predictive performance with substantially less communication (e.g., a dynamic protocol with $\sigma_{\Delta=0.7}$ reaches a performance comparable to a periodic protocol with $\sigma_{b=10}$ using only half of the communication). The more learners communicate, the lower their cumulative loss, with the serial baseline performing the best.

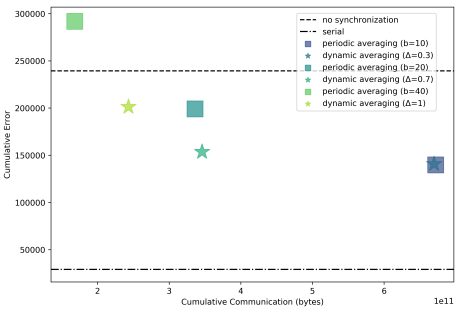


Fig. 5.1. Cumulative loss and communication of distributed learning protocols with $m = 100$ (similar to McMahan et al. [25]) learners with mini-batch size $B = 10$, each observing $T = 14000$ samples (corresponding to 20 epochs for the serial baseline).

The advantage of the dynamic protocols over the periodic ones in terms of communication is in accordance with the convex case. For large synchronization periods, however, synchronizing protocols ($\sigma_{b=40}$) have even larger cumulative loss than the nosync baseline. This behavior cannot happen in the convex case, where averaging is always superior to not synchronizing [14]. In contrast, in the non-convex case local models can converge to different local minima. Then their average might have a higher loss value than each one of the local models (as illustrated in Figure 1.1(b)).

Comparison of the dynamic averaging protocol with FedAvg: Having shown that dynamic averaging outperforms standard periodic averaging, we proceed by comparing it to a highly communication-efficient variant of periodic averaging, denoted **FedAvg** [25], which poses a state-of-the-art for decentralized deep learning under communication-cost constraints.

Using our terminology, FedAvg is a periodic averaging protocol that uses only a randomly sampled subset of nodes in each communication round. This subsampling leads to a reduction of total communication by a constant factor compared to standard periodic averaging. In order to compare dynamic averaging to FedAvg, we repeat the MNIST classification using CNNs and multiple configurations of dynamic averaging and FedAvg.

Figure 5.2 shows the evolution of cumulated communication during model training comparing dynamic averaging to the optimal configuration of FedAvg with $b = 50$ and $C = 0.3$ for MNIST (see Section 3 in McMahan et al. [25]) and variants of this configuration. We find noteworthy spreads between the communication curves, while all approaches have comparable loss. The communication amounts of all FedAvg variants increase linearly during training. The smaller the fraction of learners C involved in synchronization, the smaller the amount of com-

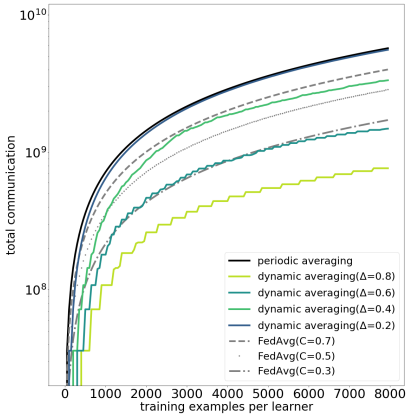


Fig. 5.2. Evolution of cumulative communication for different dynamic averaging and FedAvg protocols on $m = 30$ learners using mini-batch size $B = 10$.

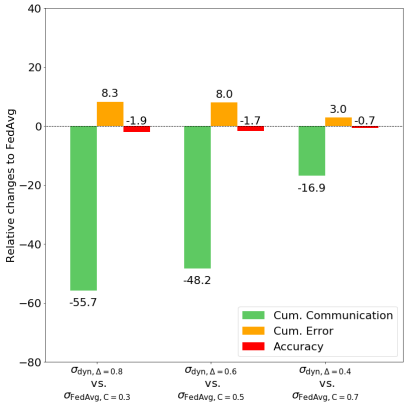
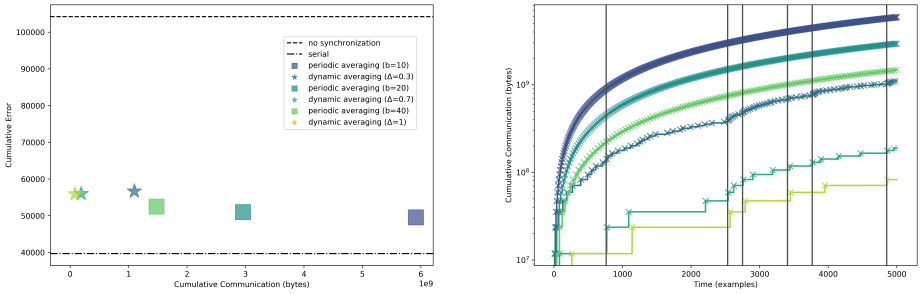


Fig. 5.3. Comparison of the best (second best, third best) performing setting of the dynamic averaging protocol with its FedAvg counterpart.

munication. In contrast, we observe step-wise increasing curves for all dynamic averaging protocols which reflect their inherent irregularity of communication. Dynamic averaging with $\Delta = 0.6$ and $\Delta = 0.8$ beat the strongest FedAvg configuration in terms of cumulative communication, the one with $\Delta = 0.8$ even with a remarkable margin. We find these improvements of communication efficiency to come at almost no cost: Figure 5.3 compares the three strongest configurations of dynamic averaging to the best performing FedSGG one, showing a reduction of over 50% in communication with an increase in cumulative loss by only 8.3%. The difference in terms of classification accuracy is even smaller, dynamic averaging is only worse by 1.9%. Allowing for more communication improves the loss of dynamic averaging to the point where dynamic averaging virtually the same accuracy as FedAvg with 16.9% less communication.

Adaptivity to Concept Drift: The advantage of dynamic averaging over any periodically communicating protocol lies in the adaptivity to the current hardness of the learning problem, measured by the in-place loss. For fixed target distributions, this loss decreases over time so that the dynamic protocol reduces the amount of communication continuously until it reaches quiescence, if no loss is suffered anymore. In the presence of concept drifts, such quiescence can never be reached; after each drift, the learners have to adapt to the new target. In order to investigate the behavior of dynamic and periodic averaging in this setting, we perform an experiment on a synthetic dataset generated by a random graphical model [4]. Concept drifts are simulated by generating a new random graphical model. Drifts are triggered at random with a probability of 0.001 per round.

Figure 5.4(a) shows that in terms of predictive performance, dynamic and periodic averaging perform similarly. At the same time, dynamic averaging re-



(a) cumulative loss and communication

(b) development of cumulative comm.

Fig. 5.4. Experiment with periodic and dynamic averaging protocols on $m = 100$ learner after training on 5000 samples per learner from a synthetic dataset with concept drifts (indicated by vertical lines).

quires up to an order of magnitude less communication to achieve it. Examining the cumulative communication over time in Figure 5.4(b), one can see that dynamic averaging communicates more after each concept drift and decreases communication until the next drift. This indicates that dynamic averaging invests communication when it is most impactful and can thereby save a substantial amount of communication in between drifts.

Case Study on Deep Driving: After having studied dynamic averaging in contrast to periodic approaches and FedAvg on MNIST and a synthetic dataset with concept drifts, we analyze how the suggested protocol performs in the realistic application scenario of in-fleet training for autonomous driving introduced in Section 1. One of the approaches in autonomous driving is direct steering control of a constantly moving car via a neural network that predicts a steering angle given an input from the front view camera. Since one network fully controls the car this approach is termed **deep driving**. Deep driving neural networks can be trained on a dataset generated by recording human driver control and corresponding frontal view [1, 10, 30].

For our experiments we use a neural network architecture suggested for deep driving by Bojarski et al. [1]. The learners are evaluated by their driving ability following the qualitative evaluation made by Bojarski et al. [1] or Pomerleau [30] as well as techniques used in the automotive industry. For that, we developed a custom loss together with experts for autonomous driving that takes into account the time the vehicle drives on track and the frequency of crossing road sidelines.

Figure 5.5 shows the measurements of the custom loss against the cumulative communication. The principal difference from the previous experiments is the evaluation of the resulting models without taking into account cumulative training loss. All the resulting models as well as baseline models were loaded to the simulator and driven with a constant speed. The plot shows that each periodic

communication protocol can be outperformed by a dynamic protocol.

Similar to our previous experiments, too little communication leads to bad performance, but for deep driving, very high communication ($\sigma_{b=10}$ and $\sigma_{\Delta=0.01}$) results in a bad performance as well. On the other hand, proper setups achieve performance similar to the performance of the serial model (e.g. dynamic averaging with $\Delta = 0.1$ or $\Delta = 0.3$). This raises the question, how much diversity is beneficial in-between averaging steps and how diverse models should be initialized. We discuss this question and other properties of dynamic averaging in the following section.

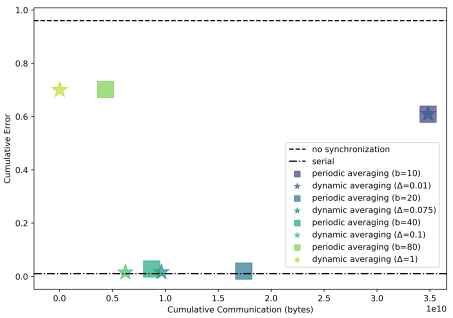


Fig. 5.5. Performance in the terms of the custom loss for the models trained according to a set of communication protocols and baseline models.

6 Discussion

A popular class of parallel learning algorithms is based on stochastic gradient descent, both in convex and non-convex learning tasks. As for all gradient-based algorithms, the gradient computation can be parallelized 'embarrassingly' [27] easily. For convex problems, the best so far known algorithm, in terms of predictive performance, in this class [34] is the distributed mini-batch algorithm [8]. For the non-convex problem of training (deep) neural networks, McMahan et al. [25] have shown that periodic averaging performs similar to the mini-batch algorithm. Section 4 substantiates these results from a theoretical perspective. Sub-sampling learners in each synchronization allows to further reduce communication at the cost of a moderate loss in predictive performance.

Note that averaging models, similar to distributed mini-batch training, requires a common architecture for all local models since the goal is to jointly train a single global model distributedly using observations from local data streams—which also sets it apart from ensemble methods.

For the convex case, Kamp et al. [17] have shown that dynamic averaging retains the performance of periodic averaging and certain serial learning algorithms (including SGD) with substantially less communication. Section 4 proves that these results are applicable to the non-convex case as well. Section 5 indicates that these results also hold in practice and that dynamic averaging indeed outperforms periodic averaging, both with and without sub-sampling of learners. This advantage is even amplified in the presence of concept drift. Moreover, dynamic averaging is a black-box approach, i.e., it can be applied with arbitrary learning algorithms (see Appendix A.5 for a comparison of using dynamic averaging with SGD, ADAM, and RMSprop).

However, averaging models instead of gradients has the disadvantage of being susceptible to outliers. That is, without a bound on the quality of local models, their average can be arbitrarily bad [34, 15]. More robust approaches are computationally expensive, though, e.g., the geometric median [9]. Others are not directly applicable to non-convex problems, e.g., the Radon point [15]. Thus, it remains an open question whether robust methods can be applied to decentralized deep learning.

Another open question is the choice of the divergence threshold Δ for dynamic averaging. The model divergence depends on the expected update steps (e.g., in the case of SGD on the expected norm of gradients and the learning rate), but the threshold is not intuitive to set. A good practice is to optimize the parameter for the desired trade-off between predictive performance and communication on a small subset of the data. It is an interesting question whether the parameter can also be adapted during the learning process in a theoretically sound way.

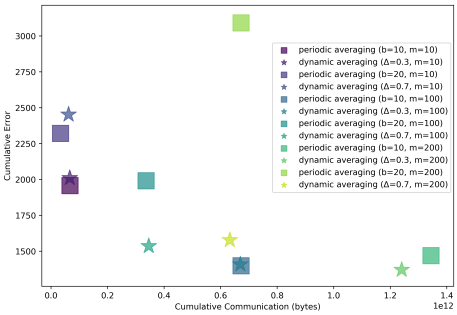
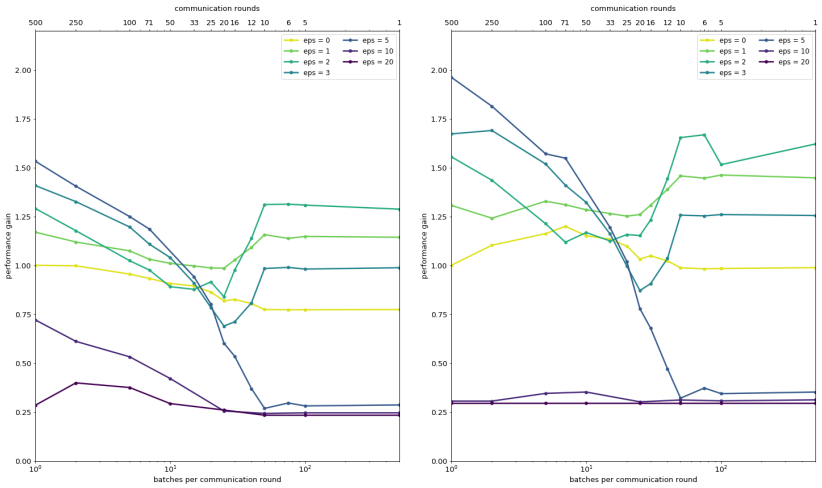


Fig. 6.1. Cumulative loss and cumulative communication of learning protocols for a different amount of learners. Training is performed on MNIST for 2, 20 and 40 epochs for $m = 10$, $m = 100$, $m = 200$ setups correspondingly.

In dynamic averaging, the amount of communication not only depends on the actual divergence of models, but also on the probability of local violations. Since the local conditions can be violated without the actual divergence crossing the threshold, these false alarms lead to unnecessary communication. The more learners in the system, the higher the probability of such false alarms. In the worst case, though, dynamic averaging communicates as much as periodic averaging. Thus, it scales at least as well as current decentralized learning approaches [25, 13]. Moreover, using a resolution strategy that tries to balance violations by communicating with just a small number of learners partially compensates for this problem. Indeed, experiments on the scalability of the approach show that dynamic averaging scales well with the number of learners (see Figure 6.1 for a comparison of using $m = 10, 100$, and 200 learners, see Appendix A.6 for details).

A general question when using averaging is how local models should be initialized. McMahan et al. [25] suggest using the same initialization for all local models and report that different initializations deteriorate the learning process when models are averaged only once at the end. Studying the transition from homogeneously initialized and converging model configurations to heterogeneously initialized and failing ones reveals that, surprisingly, for multiple rounds of averaging different initializations can indeed be beneficial. Figure 6.2 shows the



(a) static averaging protocols (b) dynamic averaging protocols

Fig. 6.2. Relative performances of averaged model obtained from various heterogeneous model initializations parameterized by ϵ (curve parameter) and various numbers of local batches b/B (abscissa). All averaged model performances are compared to an experiment with homogeneous model initializations ($\epsilon = 0$) and $b/B = 1$.

performances of dynamic and periodic averaging for different numbers of rounds of averaging and different levels of inhomogeneity in the initialization. The results confirm that for one round of averaging, inhomogeneous initialization deteriorates the learning process, but for more frequent rounds of averaging it shows that mild inhomogeneity actually improves training. For large heterogeneities, however, model averaging fails as expected. This raises an interesting question about the regularizing effects of averaging and its potential advantages over serial learning in case of non-convex objectives.

7 Conclusion

In decentralized deep learning there is a natural trade-off between learning performance and communication. Averaging models allows to achieve a high predictive performance with less communication compared to sharing data. The proposed dynamic averaging protocol achieves similarly high predictive performance yet requires substantially less communication. At the same time, it is adaptive to concept drifts. The method is theoretically sound, i.e., it retains the loss bounds of the underlying learning algorithm using an amount of communication that is bound by the hardness of the learning problem—measured by the loss of the underlying learning algorithm.

Bibliography

[1] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., et al.: End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316 (2016)

[2] Boley, M., Kamp, M., Keren, D., Schuster, A., Sharfman, I.: Communication-efficient distributed online prediction using dynamic model synchronizations. In: BD3@ VLDB. pp. 13–18 (2013)

[3] Bottou, L.: Stochastic gradient learning in neural networks. Proceedings of Neuro-Nımes 91(8), 0 (1991)

[4] Bshouty, N.H., Long, P.M.: Linear classifiers are nearly optimal when hidden variables have diverse effects. Machine Learning 86(2), 209–231 (2012)

[5] Chen, J., Monga, R., Bengio, S., Jozefowicz, R.: Revisiting distributed synchronous SGD. In: International Conference on Learning Representations Workshop Track (2016)

[6] Chollet, F., et al.: Keras. https://keras.io (2015)

[7] Dean, J., Corrado, G.S., Monga, R., Chen, K., Devin, M., Quoc, V.L., Mao, M.Z., Ranzato, M., Senior, A., Tucker, P., Yang, K., Ng, A.Y.: Large scale distributed deep networks. In: Advances in Neural Information Processing Systems. pp. 1223–1231 (2012)

[8] Dekel, O., Gilad-Bachrach, R., Shamir, O., Xiao, L.: Optimal distributed online prediction using mini-batches. Journal of Machine Learning Research 13, 165–202 (2012)

[9] Feng, J., Xu, H., Mannor, S.: Outlier robust online learning. CoRR abs/1701.00251 (2017)

[10] Fernando, T., Denman, S., Sridharan, S., Fookes, C.: Going deeper: Autonomous steering with neural memory networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 214–221 (2017)

[11] Gabel, M., Keren, D., Schuster, A.: Communication-efficient distributed variance monitoring and outlier detection for multivariate time series. In: Proceedings of the 28th International Parallel and Distributed Processing Symposium (IPDPS). pp. 37–47. IEEE (2014)

[12] Giatrakos, N., Deligiannakis, A., Garofalakis, M., Sharfman, I., Schuster, A.: Prediction-based geometric monitoring over distributed data streams. In: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. pp. 265–276. ACM (2012)

[13] Jiang, Z., Balu, A., Hegde, C., Sarkar, S.: Collaborative deep learning in fixed topology networks. In: Advances in Neural Information Processing Systems. pp. 5904–5914 (2017)

[14] Kamp, M., Boley, M., Keren, D., Schuster, A., Sharfman, I.: Communication-efficient distributed online prediction by dynamic model synchronization. In: Machine Learning and Knowledge Discovery in Databases, pp. 623–639. Springer (2014)

[15] Kamp, M., Boley, M., Missura, O., Gärtner, T.: Effective parallelisation for machine learning. In: Advances in Neural Information Processing Systems. pp. 6480–6491 (2017)

[16] Kamp, M., Boley, M., Mock, M., Keren, D., Schuster, A., Sharfman, I.: Adaptive communication bounds for distributed online learning. In: 7th NIPS Workshop on Optimization for Machine Learning (2014)

[17] Kamp, M., Bothe, S., Boley, M., Mock, M.: Communication-efficient distributed online learning with kernels. In: Machine Learning and Knowledge Discovery in Databases. pp. 805–819. Springer (2016)
[18] Keren, D., Sagy, G., Abboud, A., Ben-David, D., Schuster, A., Sharfman, I., Deligiannakis, A.: Geometric monitoring of heterogeneous streams. IEEE Transactions on Knowledge and Data Engineering 26(8), 1890–1903 (2014)
[19] Keren, D., Sharfman, I., Schuster, A., Livne, A.: Shape sensitive geometric monitoring. IEEE Transactions on Knowledge and Data Engineering 24(8), 1520–1535 (2012)
[20] Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M., Tang, P.T.P.: On large-batch training for deep learning: Generalization gap and sharp minima. In: International Conference on Learning Representations (2017)
[21] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Proceedings of the 3rd International Conference on Learning Representations (2014)
[22] Lazerson, A., Sharfman, I., Keren, D., Schuster, A., Garofalakis, M., Samoladas, V.: Monitoring distributed streams using convex decompositions. Proceedings of the VLDB Endowment 8(5), 545–556 (2015)
[23] LeCun, Y.: The mnist database of handwritten digits. http://yann. lecun. com/exdb/mnist/ (1998)
[24] Mcdonald, R., Mohri, M., Silberman, N., Walker, D., Mann, G.S.: Efficient large-scale distributed training of conditional maximum entropy models. In: Advances in Neural Information Processing Systems. pp. 1231–1239 (2009)
[25] McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics. pp. 1273–1282 (2017)
[26] McMahan, B., Ramage, D., Talwar, K., Zhang, L.: Learning differentially private recurrent language models. In: International Conference on Learning Representations (2018)
[27] Moler, C.: Matrix computation on distributed memory multiprocessors. Hypercube Multiprocessors 86(181-195), 31 (1986)
[28] Nesterov, Y.: Introductory lectures on convex optimization: A basic course, vol. 87. Springer Science & Business Media (2013)
[29] Nguyen, Q., Hein, M.: The loss surface of deep and wide neural networks. In: International Conference on Machine Learning. pp. 2603–2612 (2017)
[30] Pomerleau, D.A.: Alvinn: An autonomous land vehicle in a neural network. In: Advances in Neural Information Processing Systems. pp. 305–313 (1989)
[31] Sagy, G., Keren, D., Sharfman, I., Schuster, A.: Distributed threshold querying of general functions by a difference of monotonic representation. Proceedings of the VLDB Endowment 4(2), 46–57 (2010)
[32] Sanghavi, S., Ward, R., White, C.D.: The local convexity of solving systems of quadratic equations. Results in Mathematics 71(3-4), 569–608 (2017)
[33] Shamir, O.: Without-replacement sampling for stochastic gradient methods. In: Advances in Neural Information Processing Systems. pp. 46–54 (2016)
[34] Shamir, O., Srebro, N., Zhang, T.: Communication-efficient distributed optimization using an approximate newton-type method. In: International Conference on Machine Learning. pp. 1000–1008 (2014)
[35] Sharfman, I., Schuster, A., Keren, D.: A geometric approach to monitoring threshold functions over distributed data streams. Transactions on Database Systems 32(4) (2007)

[36] Sharfman, I., Schuster, A., Keren, D.: Shape sensitive geometric monitoring. In: Proceedings of the ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. pp. 301–310. ACM (2008)
[37] Smith, V., Chiang, C.K., Sanjabi, M., Talwalkar, A.S.: Federated multi-task learning. In: Advances in Neural Information Processing Systems. pp. 4424–4434 (2017)
[38] Tieleman, T., Hinton, G.: Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural networks for machine learning 4(2), 26–31 (2012)
[39] Verner, U., Schuster, A., Silberstein, M.: Processing data streams with hard real-time constraints on heterogeneous systems. In: Proceedings of the international conference on Supercomputing. pp. 120–129. ACM (2011)
[40] Wang, W., Srebro, N.: Stochastic nonconvex optimization with large minibatches. CoRR abs/1709.08728 (2017)
[41] Xavier Glorot, Y.B.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the International Conference on Artificial Intelligence and Statistics (2010)
[42] Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning requires rethinking generalization. In: Proceedings of the International Conference on Learning Representations (2017)
[43] Zhang, S., Choromanska, A.E., LeCun, Y.: Deep learning with elastic averaging sgd. In: Advances in Neural Information Processing Systems. pp. 685–693 (2015)
[44] Zhang, Y., Wainwright, M.J., Duchi, J.C.: Communication-efficient algorithms for statistical optimization. In: Advances in Neural Information Processing Systems. pp. 1502–1510 (2012)
[45] Zinkevich, M., Weimer, M., Smola, A.J., Li, L.: Parallelized stochastic gradient descent. In: Advances in Neural Information Processing Systems. pp. 2595–2603 (2010)

A Details on the Empirical Evaluation

In this section we provide additional details on the experimental setup and the empirical evaluation presented in Section 5.

A.1 Experiments on MNIST

For the experiments with MNIST dataset we have chosen a neural network with two convolutional layers, one max pooling layer, and two dense layers with final softmax activation. The overview of the network layers and the amount of neurons are presented in Table 1. To make the results comparable to those of McMahan et al. [25] we chose $m = 100$ learners, each training a network with roughly 10^6 weights. This distributed training is performed for three setups of both periodic and dynamic averaging (summarized in Table 2), as well as for the two baselines, serial and nosync.

The employed loss function for training is categorical crossentropy and the learning algorithm is mini-batch SGD (as it is implemented in the Keras library [6]).

Table 1 The architecture of the Convolutional Neural Network used for MNIST dataset. Printout of the parameters made via Keras library.

Layer Type	Output Shape	#Weights
Conv2D	(26, 26, 32)	320
Conv2D	(24, 24, 64)	18,496
MaxPooling2D	(12, 12, 64)	0
Dropout	(12, 12, 64)	0
Flatten	(9216)	0
Dense	(128)	1,179,776
Dropout	(128)	0
Dense	(10)	1,290
Total		1,199,882

The parameters of the learning algorithm are optimized on a separate dataset. For the serial learner, the optimal parameters are a batch size of $B = 10$ samples and a learning rate of $\eta = 0.1$. With these parameters the serial model reaches an accuracy of 0.99 (calculated as an incremental value during the online training time) which is competitive on the MNIST dataset. For the distributed learners, the batch size is again $B = 10$, but the learning rate is higher with $\eta = 0.25$.

For the distributed setups, each of the $m = 100$ learners observes 14000 examples. At the same time, the serial baseline observes the same amount of examples as the entire distributed learning system, i.e., $100 \cdot 14000$ samples of the 70000 data points. Thus, while each local learners observes only a fraction of the dataset, the serial learner trains for 20 epochs over the data. This explains the substantially lower cumulative loss of the serial baseline compared to the distributed protocols. However, the serial baseline requires centralization of all training data which is often infeasible in decentralized application scenarios. Moreover, this data needs to be centrally processed, resulting in a runtime that is roughly m-times higher than that of the distributed approaches.

Table 2 Overview of the different communication protocol configurations used for MNIST experiment. Except for protocol parameters Δ and b, all other parameters are kept constant between configurations.

protocol configurations		
type	**name**	**parameter**
periodic protocol	$\sigma_{b=10}$	$b = 10$
	$\sigma_{b=20}$	$b = 20$
	$\sigma_{b=40}$	$b = 40$
dynamic protocol	$\sigma_{\Delta=0.3}$	$\Delta = 0.3$
	$\sigma_{\Delta=0.7}$	$\Delta = 0.7$
	$\sigma_{\Delta=1}$	$\Delta = 1$

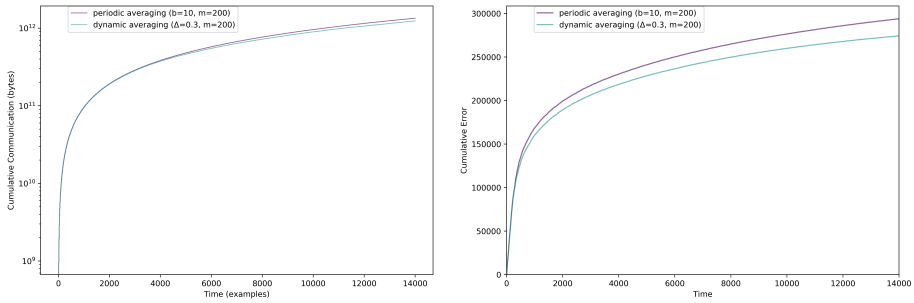


Fig. A.1. (a) The cumulative communication development during the training on MNIST dataset for 40 epochs using dynamic averaging ($\Delta = 0.3, b = 10$) and periodic averaging ($b = 10$). (b) The cumulative loss development during the training on MNIST dataset for 40 epochs with one dynamic and one periodic protocols.

As a solution, periodic averaging allows to sacrifice predictive performance to allow decentralized computation in such a way that the more the protocol communicates, the more performance is maintained. Moreover, for each setup of periodic averaging, a parameter Δ of dynamic averaging can be found such that it has similar predictive performance but requires less communication. As an example of this, in Figure A.1 we present the cumulative communication and error over time of to similarly performing protocols, $\sigma_{\Delta=0.3}$ and $\sigma_{b=10}$. During first 500 examples, when both protocols suffer a lot of loss, dynamic averaging invests more communication, resulting in faster convergence and thus a lower increase cumulative error. After the first 3000 examples, dynamic averaging reduces the amount of communication without suffering more loss. This leads eventually to both smaller cumulative error and cumulative communication.

A.2 Comparison with FedAvg

In Table 3, we provide an overview of the dynamic averaging and FedAvg protocol configurations used in the section 5. The experiment runs with $m = 30$ learners, local batches of size $B = 10$ and $b = 50$ training examples per learner between subsequent

synchronization checks. The FedAvg parameter C [25] is the fraction of learners involved in a particular model synchronization and their parameter E, the number of local batches, corresponds to the fraction of b and B, $E = b/B$. Each learner is trained on 8000 training examples.

Table 3 Overview of the different communication protocol configurations for the comparison with FedAvg. Except for protocol parameters Δ and C, all other parameters are kept constant between configurations.

protocol configurations		
type	**name**	**parameter**
periodic protocol	$\sigma_{b=50}$	–
dynamic protocol	$\sigma_{\Delta=0.1}$	$\Delta = 0.1$
	$\sigma_{\Delta=0.2}$	$\Delta = 0.2$
	$\sigma_{\Delta=0.4}$	$\Delta = 0.4$
	$\sigma_{\Delta=0.6}$	$\Delta = 0.6$
	$\sigma_{\Delta=0.8}$	$\Delta = 0.8$
FedAvg	$\sigma_{\text{FedAvg,C=0.3}}$	$C = 0.3$
	$\sigma_{\text{FedAvg,C=0.5}}$	$C = 0.5$
	$\sigma_{\text{FedAvg,C=0.7}}$	$C = 0.7$

The loss cumulated by $\sigma_{\Delta=0.6}$ and $\sigma_{\Delta=0.8}$ during training is only very slightly larger than those of the other σ_Δ and σ_{FedAvg}. Fig. A.2 illustrates this trade-off in greater detail, showing the development of cumulative communication and cumulative loss over time. Both kinds of protocols lead to a significant reduction of communication in comparison to $\sigma_{b=50}$ (see Fig. A.3). In the case of $\sigma_{\Delta=0.8}$ of more than 80%. Both, σ_Δ and σ_{FedAvg}, go along with slight increases of the cumulative loss (Fig. A.3(a)) and slight decreases of model accuracy (Fig. A.3(b)).

A.3 Adaptivity to Concept Drift

To assess the adaptivity of dynamic averaging, we compare it to periodic averaging on a synthetic binary classification dataset with samples from $\mathbb{R}^d$, with $d = 50$, generated using a random graphical model [4]. Concept drifts are triggered at random with a probability of 0.001. A concept drift is simulated by generating a new graphical model. Table 1 provides an overview of the network layers and the amount of neurons used. Figure A.4, that shows the development of cumulative loss and communication over time, confirms that dynamic averaging adapts the amount of communication: right after a concept drift the number of synchronizations (indicated by a cross mark) is high and decreases as soon as the instantaneous loss of the learners—i.e., observable as the growth of the cumulative error—decreases.

A.4 Deep Driving Experiments

The deep driving experiments is an example for an autonomous driving functionality that can be learned in-fleet. That is, every vehicle continuously trains a local model

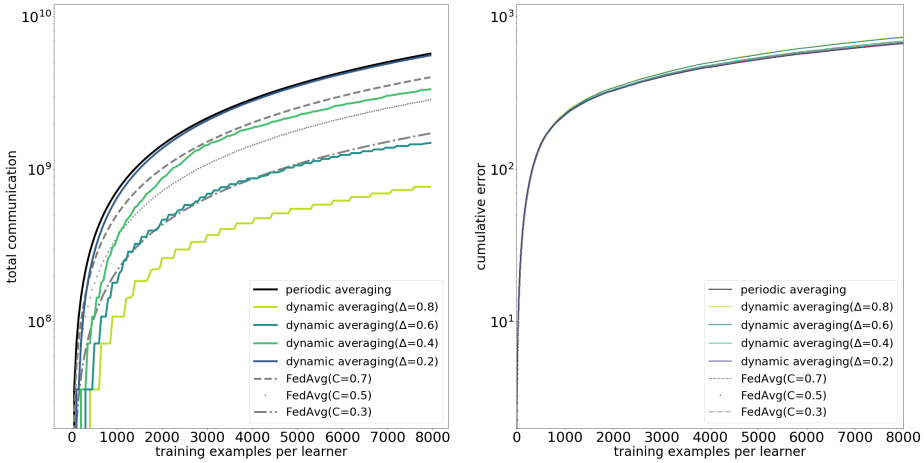


Fig. A.2. Evolution of (a) total communication and (b) cumulative error during training for different dynamic averaging and FedAvg protocols.

Table 4 Overview of the different communication protocol configurations for the analysis of concept drift. Except for protocol parameters Δ and b, all other parameters are kept constant between configurations.

protocol configurations		
periodic protocol	$\sigma_{b=10}$	$b = 10$
	$\sigma_{b=20}$	$b = 20$
	$\sigma_{b=40}$	$b = 40$
dynamic protocol	$\sigma_{\Delta=0.3}$	$\Delta = 0.3$
	$\sigma_{\Delta=0.7}$	$\Delta = 0.7$
	$\sigma_{\Delta=1}$	$\Delta = 1$

based on its observations. However, this requires to infer the correct output locally for training.

Here, deep driving is a particularly well-suited application, since the correct output for training can be inferred from a human driver by mimicking his driving behavior using a frontal view camera as input [1, 10, 30]. Using only vehicles in a specific region, the data seen by an individual local learner has in good approximation a low variability and heterogeneity, because people driving cars tend to stay close to their base. Thus, data from cars from a similar region can be assumed to be fairly homogeneously distributed. In order to mimic this scenario, we record human driving behavior in a simulation for multiple drivers on a single track.

However, data of cars from different base locations, collected at different times or in different cars will underlie different (local) approximations of the actual distribution. This actual distribution has in contrast to its local approximations a large variability and heterogeneity, even if one considers only the minor set of tasks solved by machine learning. Thus, for in-fleet training over various regions the data cannot be assumed iid

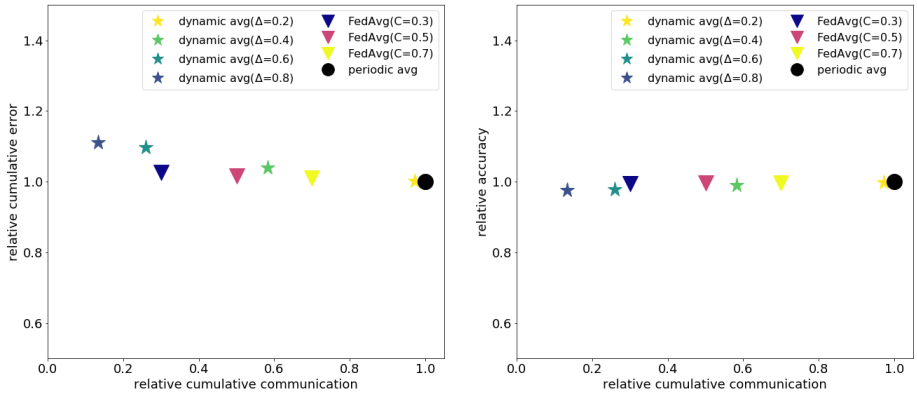


Fig. A.3. Evaluation of the dynamic averaging and FedAvg protocols relative to periodic averaging (a) in terms of cumulative error and communication, (b) in terms of accuracy and communication.

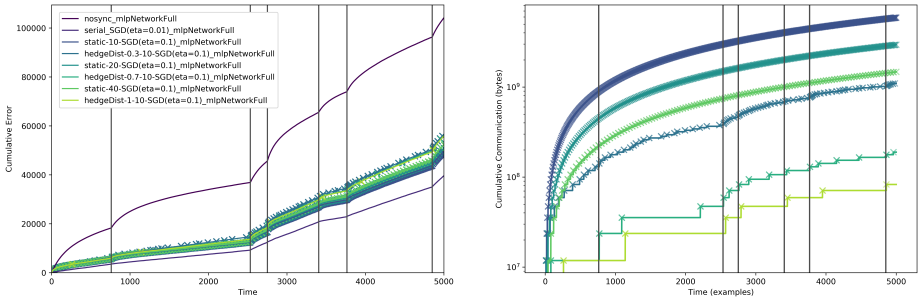
anymore, but empirical results [25] and recent extensions to the federated learning [37] suggest that the approach is capable of handling non-iid data. Analyzing these and other challenges for in-fleet learning are an interesting direction for future work.

In the following, we provide details on the experimental setup. The architecture of the layers of the deep driving network can be seen in Table 5. The employed loss func-

Table 5 The architecture of the Convolutional Neural Network used for deep driving. Printout of the parameters made via Keras library.

Layer Type	Output Shape	#Weights
Conv2D	(32, 158, 24)	1,824
Conv2D	(14, 77, 36)	21,636
Conv2D	(5, 37, 48)	43,248
Conv2D	(3, 35, 64)	27,712
Conv2D	(1, 33, 64)	36,928
Flatten	(2112)	0
Dense	(100)	211,300
Dense	(50)	5,050
Dense	(10)	510
Dense	(1)	11
Total		348,219

tion is squared error as it is implemented in Keras "mean_squared_error". The network is optimized using Stochastic Gradient Descent. The experiments were run with train-



(a) cumulative loss (b) cumulative communication

Fig. A.4. Development of cumulative loss and communication during training on 5000 examples per node from a synthetic dataset with concept drifts (indicated by vertical lines).

ing batch size of $B = 10$ and learning rate $\eta = 0.1$ both for the baselines and local learners. The input to the network is the front camera view from a car driven in a simulator[8]. The output of the network is a steering angle that allows to control the car in the autonomous mode in the simulator when the speed is kept on a constant level.

During the evaluation a trained model has been loaded to drive the car in the autonomous regime in the simulator. The time that the model is able to control the car without going off the road or crashing is measured. The longest time t_{max} is the time for the model that is able to keep going for 2 laps on the track or the maximum time of all the models in one experiment. Also the amount of times the car has touched the sideline of the road is counted together with the time duration while car is still on the sideline t_{line}. Then a frequency c of sideline crossings is calculated in a form $\frac{\#crossings}{t}$, where t is the time before going off road or crash, and the maximal frequency among all the models is assigned to c_{max}. The overall formula for the custom loss is:

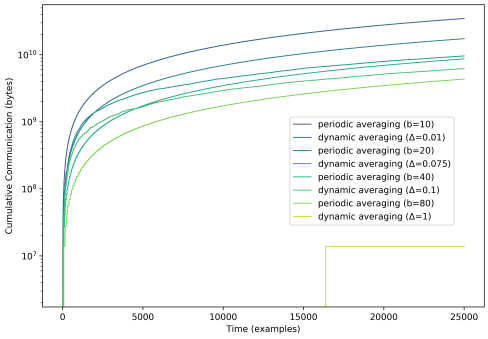


Fig. A.5. Evaluation of cumulative communication for the deep driving experiment.

$$L_{dd} = \lambda \frac{t_{max} - t}{t_{max}} + \mu \frac{c}{c_{max}} + (1 - \mu - \lambda) \frac{t_{line}}{t}$$

where t is the time that the model is able to drive on the road, $\lambda, \mu \in [0; 1]$ are weighting coefficients. For the experiment $\lambda = 0.8$ and $\mu = 0.15$ were used. The amount of examples shown to each of the $m = 10$ learners is 25000, i.e., the dataset was shown

[8] https://github.com/udacity/self-driving-car-sim

to the serial learner approximately 5 times (the overall size of the dataset is ≈ 48000). The list of the considered communication protocols can be seen from Table 6.

In Section 5 we have shown that dynamic and periodic averaging can achieve similar driving performance as the serial baseline. For that, dynamic averaging requires substantially less communication than periodic. Examining the evolution of cumulative communication (see Figure A.5) shows that—similar to the results on MNIST—dynamic averaging invests a large amount of communication in the beginning and then considerably reduces communication. This pattern is most apparent for dynamic averaging with $\Delta = 0.1$.

Table 6 Overview of the different communication protocol configurations used for deep driving experiment. Except for protocol parameters Δ and b, all other parameters are kept constant between configurations.

protocol configurations		
type	**name**	**parameter**
periodic protocol	$\sigma_{b=10}$	$b = 10$
	$\sigma_{b=20}$	$b = 20$
	$\sigma_{b=40}$	$b = 40$
	$\sigma_{b=80}$	$b = 80$
dynamic protocol	$\sigma_{\Delta=0.01}$	$\Delta = 0.01$
	$\sigma_{\Delta=0.05}$	$\Delta = 0.05$
	$\sigma_{\Delta=0.1}$	$\Delta = 0.1$
	$\sigma_{\Delta=0.3}$	$\Delta = 0.3$

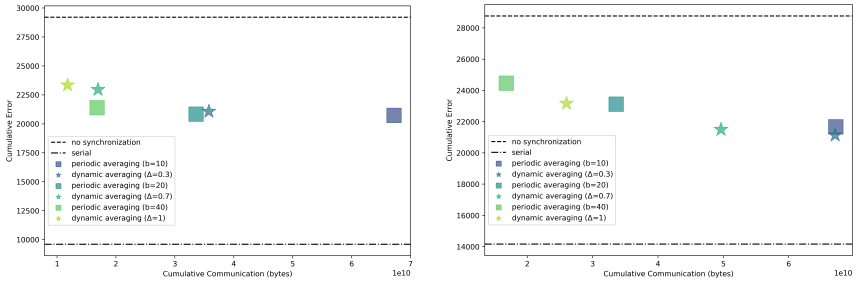
A.5 Black-Box Algorithms Evaluation

In comparison with mini-batch SGD [8, 5] our approach allows to treat the optimization algorithm as a black-box, i.e., dynamic synchronization protocols achieve performance similar to $\Pi = (\varphi^{SGD}, \sigma_{\Delta,b})$ with various different optimization algorithms. Particularly, we conducted experiments with the ADAM optimizer [21] and RMSprop [38]. The results show that the advantage of dynamic averaging over periodic also holds for those learning algorithms (see Figure A.6).

A.6 Scale-out Experiments

Communication size grows when the amount of the local learners is becoming larger, while it is still bounded for periodic and dynamic synchronization protocols (Section 4). In order to see the behavior of periodic and dynamic synchronization while changing the number of learners, we executed an experiment with setups $m = 10$, $m = 100$ and $m = 200$. The same synchronization operators were used in all three setups and the same number of examples was presented to each of the learners.

When the amount of examples per learner is fixed in order to have comparable by quality learners, the larger number of synchronizing models leads to a larger training dataset. As a consequence, it leads to a better performance of all of the setups. When



(a) φ^{Adam} (b) $\varphi^{Rmsprop}$

Fig. A.6. The averaged loss and the cumulative communication of the synchronization protocols for $m = 10$ learners. Training is performed on MNIST for 2 epochs.

the learners are saturated due to the long training and do not differ significantly enough to trigger the local conditions (see Section 3), the advantage of the dynamic protocols over the periodic ones becomes more pronounced.

The plot depicted in Figure A.7 shows the performance of two dynamic and two periodic protocols for the three scaling setups. In order to make the cumulative loss comparable it was divided by the number of learners, i.e., the cumulative loss of $m = 100$ learners is the sum of cumulative losses of all of them that is 10 times more than the sum for $m = 10$ learners—thus the first sum divided by 100 is comparable to the second one divided by 10. The plot shows that in the setup with $m = 10$ learners $\sigma_{\Delta=0.7}$ shows a comparable result to $\sigma_{b=20}$ by the price of the same communication. At the same time with $m = 100$ learners it already reaches much smaller cumulative loss. With $m = 200$ learners $\sigma_{\Delta=0.3}$ requires less communication than $\sigma_{b=10}$ that was not the case for the previous setups that empirically shows advantage of dynamic synchronization for a large dataset setup.

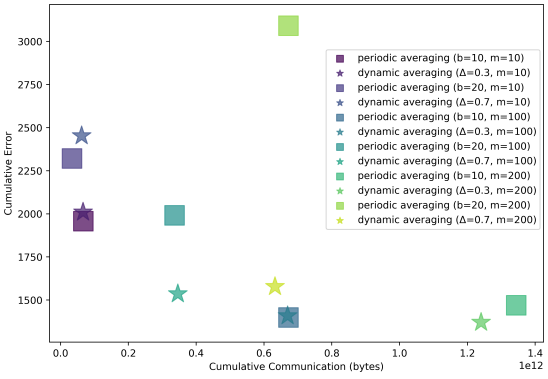


Fig. A.7. The cumulative loss and the cumulative communication of the same synchronization protocols for a different amount of learners. Training is performed on MNIST for 2, 20 and 40 epochs for $m = 10$, $m = 100$, $m = 200$ setups correspondingly.

The general approach of using local conditions to communicate efficiently has been successfully applied to massively distributed data streams [39, 31, 12]. In the worst case, though, local conditions are violated in every round. Then dynamic averaging communicates as much as periodic averaging, but even then it scales at

least as well as current decentralized learning approaches [25, 13]. E.g., FedAvg [25] performs periodic averaging on a (random) fraction $C \in (0, 1]$ of the m learners. Therefore, in the worst case dynamic averaging on m learners requires as much communication as FedAvg on m/C learners. In practice, it requires substantially less communication even on the same number of learners.

A.7 Stability of the Dynamic Averaging Protocol Regarding Model Initializations

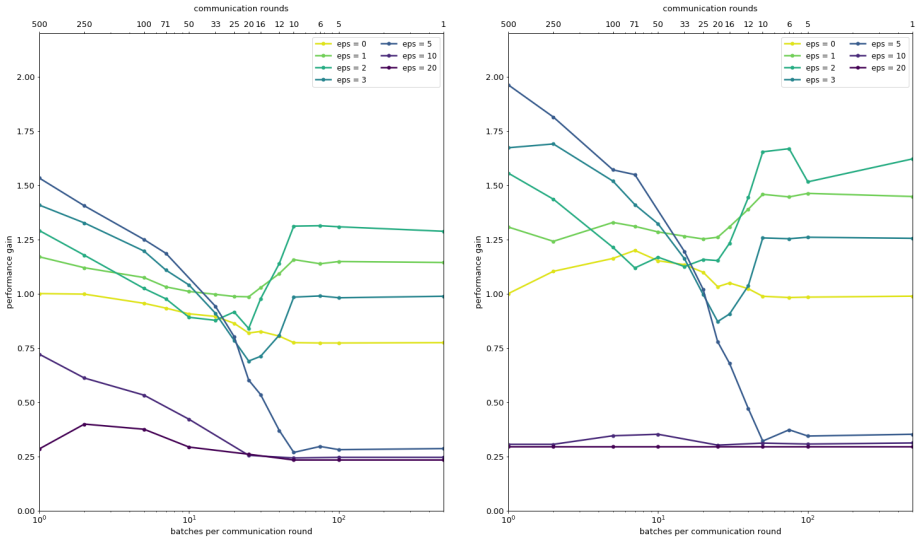


Fig. A.8. Relative performances of averaged model obtained from experiments with various heterogeneous model initializations parameterized by ϵ (curve parameter) and various numbers of local batches b/B (abscissa). All averaged model performances are compared to an experiment with homogeneous model initializations ($\epsilon = 0$) and $b/B = 1$. (a) shows results for periodic averaging, (b) for dynamic averaging.

On MNIST, naive averaging of fully trained models fails if these models are initialized differently (cp. [25]). We study the transition from homogeneously initialized and converging learning configurations to heterogeneously initialized and failing models to better understand how the regularizing effects of model averaging impact the quality of the averaged model. Failing model averaging is due to distances between the weight vectors of the different learners which are large compared to the scale of local convexity around local minima. In decentralized deep learning, there are several potential sources for such large distances. Two important ones are large numbers of local mini-batches, b/B, within one communication round and heterogeneous initializations. Further parameters, especially the learning rate, are important as well, however, their not specifically for the problem of decentralized learning.

To parameterize the transition from homogeneous to heterogeneous initializations, we start with a homogeneous initialization according to Xavier Glorot [41] and impose noise at different scales ϵ on the homogeneous initialization. The noise scale ϵ is measured relative to the scale of the homogeneous initialization. We conduct experiments with $m = 10$ learners, $B = 10$ and 500 training examples per learner for a grid of ϵ and b/B combinations. In Fig. A.8, each point corresponds to the average model performance after running one such experiment. The b/B dependency of the averaged model performance is shown on the abscissa, noise scale ϵ serves as curve parameter. Fig. A.8(a) shows the results for periodic averaging, Fig. A.8 (a) for dynamic model averaging. The averaged model performances are not shown as absolute values but relative to the configuration with $\epsilon = 0$ and $b/B = 1/10$ which corresponds to homogeneously initialized models which communicate after processing one mini-batch.

For $\epsilon = 0$, i.e. homogeneously initialized models, we find a weak dependence of resulting model performance on the number of local mini-batches. Even configurations with very large numbers of local batches between two subsequent model averagings lead to converging averaged models (cp. [25]). However, this finding can be extended to the heterogeneous case, if the scales of these heterogeneities are at the scale of the underlying homogeneous initialization, $\epsilon = 1, 2, 3$. For large heterogeneities, however, model averaging fails as expected, e.g. for $\epsilon = 20$. The transition between these two regimes occurs between $\epsilon = 5$ and $\epsilon = 10$, which show a strong dependency of model convergence on the number of local mini-batches. This critical scale of heterogeneity imposes a constrain on the choice of the dynamic protocol parameter Δ which indirectly determines the average distances between the different weight vectors before model averaging.

B Details on Theory

In the following we prove that for the base learning algorithm stochastic gradient descent (SGD), dynamic averaging for deep neural networks retains the regret bound of periodic averaging. Before proving this result, we introduce the notion of regret. The cumulative loss with respect to a reference model is denoted **regret**

$$R(T, m) = \sum_{t=1}^{T} \sum_{i=1}^{m} \ell_t^i(f_t^i) - \ell_t^i(f^*) \ .$$

Guarantees are given by a **regret bound**, i.e., for all reference models $f^* \in \mathcal{F}$ and all sequences of losses it holds that the regret is smaller than the regret bound $\mathbf{R}(T, m)$ (e.g, for serial Stochastic Gradient Descent φ^{SGD} with convex loss functions the regret is bounded by $\mathcal{O}(\sqrt{T})$, for a non-synchronizing decentralized learning system using $m \in \mathbb{N}$ learners and φ^{SGD}, the regret is bounded by $\mathcal{O}(\sqrt{mT})$ [28]).

We now show that, given two model configurations $\mathbf{d}$ and $\mathbf{s}$, applying the dynamic averaging operator σ_Δ to $\mathbf{d}$ and the static averaging operator σ_b to $\mathbf{s}$ increases their average squared pairwise model distances by at most Δ.

Lemma 4 (Kamp et al. [17]) *Let $\mathbf{d}_t, \mathbf{s}_t \in \mathcal{F}^m$ be model configurations at time $t \in \mathbb{N}$. Then*

$$\frac{1}{m} \sum_{i=1}^{m} \|\sigma_\Delta(\mathbf{d}_t)^i - \sigma_b(\mathbf{s}_t)^i\|^2 \leq \frac{1}{m} \sum_{i=1}^{m} \|d_t^i - s_t^i\|^2 + \Delta \ .$$

Proof. We consider the case $t \mod b = 0$ (otherwise the claim follows immediately). Expressing the pairwise squared distances via the difference to $\overline{d_t}$ and using the definitions of σ_b and σ_Δ we can bound

$$\frac{1}{m}\sum_{i=1}^{m}\|\sigma_\Delta(\mathbf{d}_t)^i - \sigma_b(\mathbf{s}_t)^i\|^2 = \frac{1}{m}\sum_{i=1}^{m}\|\sigma_\Delta(\mathbf{d}_t)^i - \overline{d_t} + \overline{d_t} - \overline{s_t}\|^2$$

$$= \underbrace{\frac{1}{m}\sum_{i=1}^{m}\|\sigma_\Delta(\mathbf{d}_t)^i - \overline{d_t}\|^2}_{\leq \Delta, \text{ by (ii) of Def. 2}} + 2\underbrace{\langle\frac{1}{m}\sum_{i=1}^{m}\sigma_\Delta(\mathbf{d}_t)^i - \overline{d_t}, \overline{d_t} - \overline{s_t}\rangle}_{=0, \text{ by (i) of Def. 2}} + \|\overline{d_t} - \overline{s_t}\|^2$$

$$\leq \Delta + \|\frac{1}{m}\sum_{i=1}^{m}(d_t^i - s_t^i)\|^2 = \Delta + \frac{1}{m}\sum_{i=1}^{m}\|d_t^i - s_t^i\|^2 \ .$$

□

Using this, we provide the proof of Theorem 2 in Boley et al. [2]. For that, we first recapitulate the Theorem.

Theorem 5 (Boley et al. [2]) *Let ℓ be an L-Lipschitz loss function and the update rule φ be a contraction with constant $c \in \mathbb{R}$. Then, for batch sizes $b \geq \log_2^{-1} c^{-1}$ and divergence thresholds $\Delta \leq \epsilon/2L$, the average regret of using a partial synchronization operator σ_Δ instead of σ_b is bounded by ϵ, i.e., for all rounds $t \in \mathbb{N}$ it holds that the average regret $1/m\sum_{i=1}^{m}|\ell_t^i(d_t^i) - \ell_t^i(s_t^i)|$ is bounded by ϵ where d and s denote the models at learner i and time t maintained by σ_Δ and σ_b, respectively.*

Proof. We proof the claim within two steps. First we note that a regret bound is induced by a bound on the average distances between pairs of models at the local nodes. Then we show that such a bound is retained between the local model pairs resulting from static and dynamic synchronization. Using the Lipschitz continuity of ℓ we can see that the average regret at round t is bounded as $\left|\ell_t^i(d_t^i) - \ell_t^i(s_t^i)\right| \leq L\|d_t^i - s_t^i\|$. Hence, for the desired average regret bound of ϵ it is sufficient to show that at all times $t \in \mathbb{N}$ it holds that the average pair-wise model distance at the local learners is bounded by $2\Delta = \epsilon/(L)$, i.e.,

$$\frac{1}{m}\sum_{l=1}^{k}\|d_t^i - s_t^i\| \leq 2\Delta \ . \tag{3}$$

We now show that Eq. (3) is retained throughout all rounds $t \in \mathbb{N}$. Before the first synchronization, i.e., for $t \leq b$, both weight sequences are identical and the bound holds. Moreover, if $t-1$ is not a synchronization step, i.e., $t-1 \mod b \neq 0$, the bound is preserved for $\mathbf{d}_t$ and $\mathbf{s}_t$ due to φ being a contraction. Hence, the crucial case is $t > b$ with $(t-1) \mod b = 0$. Using Lemma 4 we get that

$$\frac{1}{m}\sum_{i=1}^{m}\|\sigma_\Delta(\mathbf{d}_t)^i - \sigma_b(\mathbf{s}_t)^i\|^2 \leq \underbrace{\frac{1}{m}\sum_{i=1}^{m}\|d_t^i - s_t^i\|^2}_{(*)} + \Delta \ .$$

Applying the contraction property of φ on $(*)$ yields

$$\frac{1}{m}\sum_{i=1}^{m}\|d_t^i - s_t^i\|^2 \leq c^b \underbrace{\frac{1}{m}\sum_{i=1}^{m}\|d_{t-b}^i - s_t^i\|^2}_{\leq \Delta \text{ by IH}} \leq c^b\Delta + \Delta \leq 2\Delta \ .$$

The last inequality follows from the fact that $b \geq \log_2^{-1} c^{-1} = \log_c 1/2$. □

It follows that if the assumptions of Theorem 5 hold, then dynamic averaging retains the regret bound of periodic averaging.

It remains to show that SGD is a contraction. For convex loss functions, one can show [45] that it is a contraction for sufficiently small constant learning rates: for $\eta \leq (\rho L + \lambda)^{-1}$ the updates do contract with constant $c = 1 - \eta\lambda$. Here, $\eta, \lambda \in \mathbb{R}_+$ denote the learning rate and regularization parameter, $L \in \mathbb{R}_+$ the Lipschitz constant of the loss function, and $\rho \in \mathbb{R}_+$ the data radius (i.e., for all $x \in X \|x\|_2 \leq \rho$).

This also holds for the non-convex case if for any round $t \in T$ the loss function is locally convex in a bounded region around the span of the models in $\mathbf{d}_t$ and $\mathbf{s}_t$. Since the distance of all models in $\mathbf{d}_t$ to the models $\mathbf{s}_t$ is bounded by 2Δ and the distance of all models in $\mathbf{d}_t$ to their average $\overline{d_t}$ is bounded by Δ, all models lie in a 2Δ bounded region around $\overline{d_t}$. If moreover the update magnitude is bounded by R, then all models remain in a $2\Delta R$ bounded region. Since the update magnitude for SGD is given by the norm of the gradient and the learning rate, the update magnitude can be bounded by $R = \eta L$. Thus for all models, if in any round $t \in \mathbb{N}$ the loss function is locally convex in a $2\Delta R$-radius around $\overline{d_t}$, then SGD is a contraction. To see that this is a non-trivial but realistic assumption, see Sanghavi et al. [32], Wang and Srebro [40] on local convexity and Nguyen and Hein [29], Keskar et al. [20] on the the loss surface of deep learning. It follows that under these assumptions, dynamic averaging with SGD retains the regret bound of periodic averaging.

Corollary 6 *Let ℓ be an L-Lipschitz loss function, $\lambda \in \mathbb{R}$ a regularization parameter, $\eta \leq (\rho L + \lambda)^{-1}$ a learning rate, $\rho \in \mathbb{R}_+$ the data radius, $b \geq \log_2^{-1} c^{-1}$ the batch sizes and $\Delta \leq \epsilon/2L$ the divergence threshold. , dynamic averaging retains the regret bound of periodic averaging. In round $t \in \mathbb{N}$, let $\mathbf{d}_t$ and $\mathbf{s}_t$ denote the models maintained by dynamic and periodic averaging, respectively. If in any round $t \in \mathbb{N}$ the loss function is locally convex in a $2\Delta R$-radius around $\overline{d_t}$, then dynamic averaging with stochastic gradient descent retains the regret bound of periodic averaging.*

Together with Proposition 3 it follows that dynamic averaging is consistent as in Definition 1.

C Federated Learning: Different Sampling Rates and non-iid Data

In order to analyze the proposed approach theoretically, we used a strict notion with fixed, balanced sampling rate and iid data. In contrast, McMahan et al. [25] introduced federated learning as a learning task with (i) non-iid data, (ii) unbalanced data sampling rates, (iii) massively distributed systems, and (iv) limited communication infrastructure.

Naturally, dynamic averaging is well suited for massively distributed systems with limited communication infrastructure—in the worst case it performs similar to periodic averaging, in the best case it has similar predictive performance with orders of magnitude less communication. Regarding non-iid data, McMahan et al. [25] provide empirical indications that differences in local data distributions do not significantly deteriorate the learning process. While this should hold for dynamic averaging as well,

Algorithm 2: Dynamic Averaging Protocol for Unbalanced Data

Input: divergence threshold Δ, batch size b

Initialization:
local models $f_1^1, \ldots, f_1^m \leftarrow$ one random f
reference vector $r \leftarrow f$
violation counter $v \leftarrow 0$

Round t **at node** i**:**
observe $E_t^i \subset X \times Y$ with $\left|E_t^i\right| = B^i$
update f_{t-1}^i using the learning algorithm φ
if $t \mod b = 0$ **and** $\|f_t^i - r\|^2 > \Delta$ **then**
send f_t^i and B^i to coordinator (violation)

At coordinator on violation:
let $\mathcal{B}$ be the set of nodes with violation
$v \leftarrow v + |\mathcal{B}|$
if $v = m$ **then** $\mathcal{B} \leftarrow [m]$, $v \leftarrow 0$
$N \leftarrow \sum_{i \in \mathcal{B}} B^i$
while $\mathcal{B} \neq [m]$ **and** $\left\| \frac{1}{N} \sum_{i \in \mathcal{B}} B^i f_t^i - r \right\|^2 > \Delta$ **do**
augment $\mathcal{B}$ by augmentation strategy
receive models from nodes added to $\mathcal{B}$
$N \leftarrow \sum_{i \in \mathcal{B}} B^i$
send model $\overline{f} = \frac{1}{N} \sum_{i \in \mathcal{B}} B^i f_t^i$ to nodes in $\mathcal{B}$
if $\mathcal{B} = [m]$ also set new reference vector $r \leftarrow \overline{f}$

further research is required to substantiate this claim, both for federated learning and dynamic averaging (cf. Smith et al. [37]).

In order to tackle the problem of unbalanced sampling rates, a simple modification of the dynamic averaging protocol is required. For that, assume that each local learner $i \in [m]$ observes $B^i \in \mathbb{N}$ samples.

We can account for the different sampling rates in the training process by using a weighted model averaging, where the weight of each model depends on the number of observed examples. Let $N = \sum_{i=1}^m B^i$ be the total number of samples observed in each round. Then, the weighted average of models is given by

$$\overline{f} = \frac{1}{N} \sum_{i=1}^{m} B^i f_t^i \ .$$

Note that this can be generalized analogously to time-dependent sampling rates $B_t^i ound$. Using this weighted average, the dynamic averaging protocol for unbalanced data is given in Algorithm 2.