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, Sankt Augustin, Germany
{michael.kamp,linara.adilova,joachim.sicking,
tim.wirtz,stefan.wrobel}@iais.fraunhofer.de
[2] Fraunhofer Center for Machine Learning, Sankt Augustin, Germany
[3] University of Bonn, Bonn, Germany
{kamp,wrobel}@cs.uni-bonn.de
[4] Volkswagen Group Research, Wolfsburg, Germany
{fabian.huger,peter.schlicht}@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. Code related to this paper is available at: https://bitbucket.org/Michael_Kamp/decentralized-machine-learning.

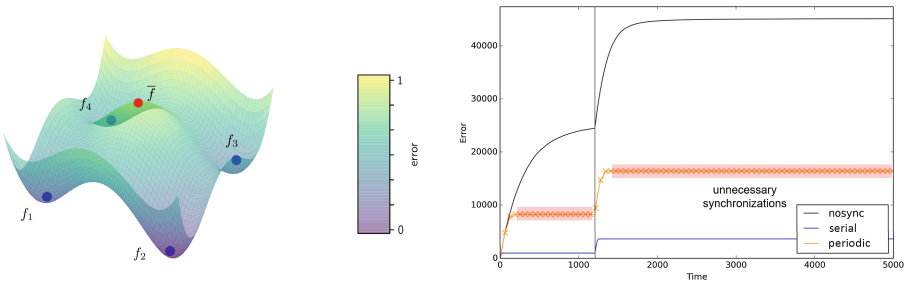
1 Introduction

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 [7], instead of raw data. This mini-batch

M. Kamp, L. Adilova and J. Sicking—These authors contributed equally.

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-10925-7_24) contains supplementary material, which is available to authorized users.

M. Berlingerio et al. (Eds.): ECML PKDD 2018, LNAI 11051, pp. 393–409, 2019.
https://doi.org/10.1007/978-3-030-10925-7_24



(a) (b)

Fig. 1. (a) 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 larger error than each of them. (b) 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.

approach performs well on tightly connected distributed systems [5,6,33] (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 [35] and average models periodically, instead of sharing gradients. Averaging models has three major advantages: (i) sending only the model parameters instead of a set of data samples reduces communication[1]; (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 [21,27,34]. 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 Fig. 1(a). For the particular case of deep learning, McMahan et al. [22] 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 so frequently that they generate an immense amount of unnecessary communication in-between drifts.

[1] 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.

In Kamp et al. [12] 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 Fig. 1(b)); 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 Sect. 2. This approach, denoted **dynamic averaging**, was proposed for online learning convex objectives [12,15]. We adapt dynamic averaging to the non-convex objectives of deep learning in Sect. 3.

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, Sect. 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** [22], a state-of-the-art approach for decentralized deep learning—more recent approaches are interesting from a theoretical perspective but show no practical improvement [11], or tackle other aspects of federated learning, such as non-iid data [31] or privacy aspects [23].

Section 6 discusses properties and limitations of dynamic averaging and puts it into context of related work, followed by a conclusion in Sect. 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 $\ell_t^i\left(f_t^i\right) = \sum_{(x,y)\in E_t^i} \ell\left(f_t^i, x, y\right)$[2]. The goal of decentralized learning is to minimize the **cumulative loss** up to a time horizon $T \in \mathbb{N}$, i.e.,

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

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** $\sigma\colon \mathcal{F}^m \to \mathcal{F}^m$ 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** $\sigma(\mathbf{f}_t)$ which replaces the local models. We measure the performance of the operator in terms of communication by the **cumulative communication**, i.e.,

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

where $c : \mathcal{F}^m \to \mathbb{N}$ 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 [21,22,27,34,35], 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 Sect. 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 the predictive performance and cumulative communication. In order to assess the efficiency of decentralized learning protocols in terms of the trade-off between loss and communication, Kamp et al. [14] introduced two criteria: consistency and adaptiveness.

Definition 1 (Kamp et al. [14]). *A distributed online learning protocol $\Pi = (\varphi, \sigma)$ processing mT inputs is **consistent** if it retains the loss of the serial online learning algorithm φ, i.e.,*

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

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

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

[2] 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.

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 [7,35], 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 Sect. 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 [15] 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.,

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

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. [12]). *A **dynamic averaging operator** with positive divergence threshold $\Delta \in \mathbb{R}_+$ and batch size $b \in \mathbb{N}$ is a synchronization operator $\sigma_{\Delta,b}$ such that $\sigma_{\Delta,b}(\mathbf{f}_t) = \mathbf{f}_t$ if $t \mod b \neq 0$ and otherwise: (i) $\overline{f}_t = \overline{\sigma_{\Delta,b}(\mathbf{f}_t)}$, i.e., it leaves the mean model invariant, and (ii) $\delta\left(\sigma_{\Delta,b}(\mathbf{f}_t)\right) \leq \Delta$, 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** $\mathcal{D} = (\varphi, \sigma_{\Delta,b})$ synchronizes the local learners using the dynamic averaging operator $\sigma_{\Delta,b}$. 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** $r \in \mathcal{F}$ [30] that is common among all learners (see [10,16,17,19,29] 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

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 $r \leftarrow \overline{f}$

divergence does not exceed the threshold, i.e., $\delta(\mathbf{f}_t) \leq \Delta$ [12, 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 equivalent to a full synchronization and the reference vector is 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 Definition 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 a 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., for all $t \in \mathbb{N}$, the synchronization operator is given by $\sigma_1(\mathbf{f}_t) = \left(\overline{f}_t, \ldots, \overline{f}_t\right)$. As base learning algorithm we use mini-batch SGD algorithm $\varphi_{B,\eta}^{\text{mSGD}}$ [7] with mini-batch size $B \in \mathbb{N}$ and learning rate $\eta \in \mathbb{R}_+$ 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 the continuous averaging protocol using mini-batch SGD. 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.

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

$$= \frac{1}{m} mf - \frac{1}{m} \eta \sum_{i=1}^{m} \sum_{j=1}^{B} \nabla \ell^{(i-1)B+j}(f) = f - \frac{1}{m} \eta \sum_{j=1}^{mB} \nabla \ell^j(f) = \varphi_{mB,\eta/m}^{\text{mSGD}}(f)$$

□

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 [18,25]—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 in the supplementary material 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. [14] shows that the cumulative communication of the dynamic averaging protocol using SGD and a divergence threshold Δ is bounded by

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

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, $c(\mathbf{f})$ 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. To emphasize the theoretical result from Sect. 4, we show that dynamic averaging indeed retains the performance of periodic averaging with substantially less communication. This is followed by a comparison of our approach with a state-of-the-art communication approach. 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 $\varphi_{B,\eta}^{\text{mSGD}}$ as learning algorithm, since recent studies indicate that it is particularly suited for training deep neural networks [32]. That is, we consider communication protocols $\Pi = (\varphi_{B,\eta}^{\text{mSGD}}, \sigma)$ 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 in the supplementary material.

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 [20]. 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**.

Figure 2 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 $\sigma_{b=1}$ using only half of the communication). The more learners communicate, the lower their cumulative loss, with the serial baseline performing the best.

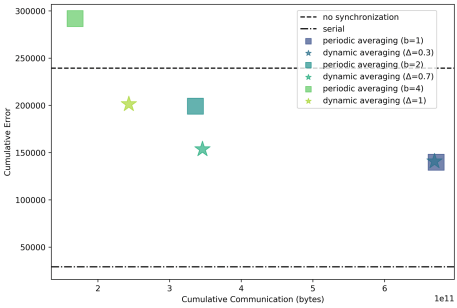


Fig. 2. Cumulative loss and communication of distributed learning protocols with $m = 100$ (similar to McMahan et al. [22]) learners with mini-batch size $B = 10$, each observing $T = 14000$ samples (corresponding to 20 epochs for the serial baseline).

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 ($\sigma_{b=4}$) 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 [12]. 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 Fig. 1(a)).

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** [22], 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 3 shows the evolution of cumulative communication during model training comparing dynamic averaging to the optimal configuration of FedAvg

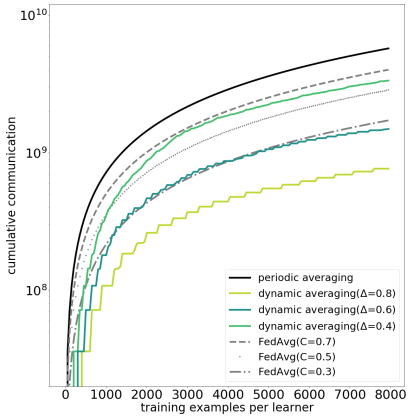


Fig. 3. Evolution of cumulative communication for different dynamic averaging and FedAvg protocols on $m = 30$ learners using a mini-batch size $B = 10$.

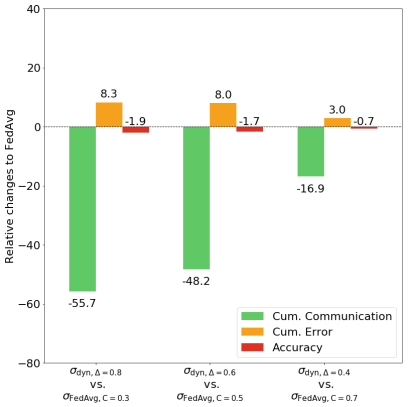
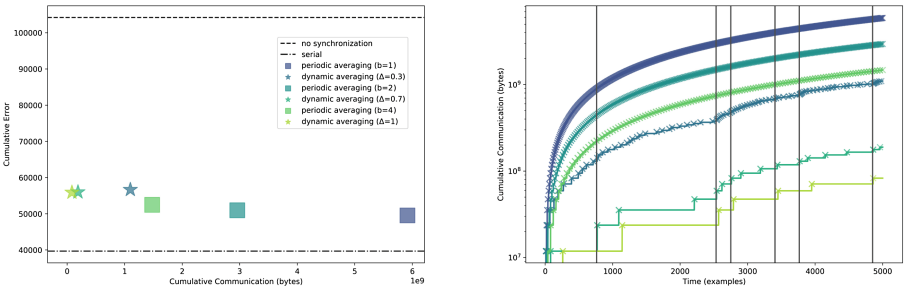


Fig. 4. Comparison of the best performing settings of the dynamic averaging protocol with their FedAvg counterparts.

with $b = 5$ and $C = 0.3$ for MNIST (see Sect. 3 in McMahan et al. [22]) and variants of this configuration. We find noteworthy spreads between the communication curves, while all approaches have comparable losses. The communication amounts of all FedAvg variants increase linearly during training. The smaller the fraction of learners, $C \in (0, 1]$, involved in synchronization, the smaller the amount of communication. 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: Fig. 4 compares the three strongest configurations of dynamic averaging to the best performing FedAvg ones, 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 has 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.



(a) cumulative loss and communication (b) cumulative communication

Fig. 5. 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 in (b)).

Figure 5(a) shows that in terms of predictive performance, dynamic and periodic averaging perform similarly. At the same time, dynamic averaging requires up to an order of magnitude less communication to achieve it. Examining the cumulative communication over time in Fig. 5(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 Sect. 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 [1,9,26].

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 [26] 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 6 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 ($\sigma_{b=1}$ 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.

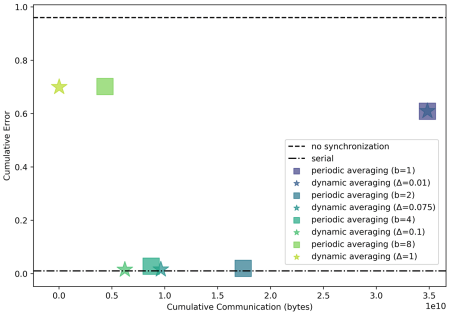


Fig. 6. Performance in the terms of the custom loss for the models trained according to a set of communication protocols and baseline models.

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' [24] easily. For convex problems, the best so far known algorithm, in terms of predictive performance, in this class [28] is the distributed mini-batch algorithm [7]. For the non-convex problem of training (deep) neural networks, McMahan et al. [22] 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. [15] 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 drifts. Additionally, dynamic averaging is a black-box approach, i.e., it can be applied with arbitrary learning algorithms (see Appendix A.5 in the supplementary material 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 [13,28]. More robust approaches are computationally expensive, though, e.g., the geometric median [8]. Others are not directly applicable to non-convex problems, e.g., the Radon point [13]. 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.

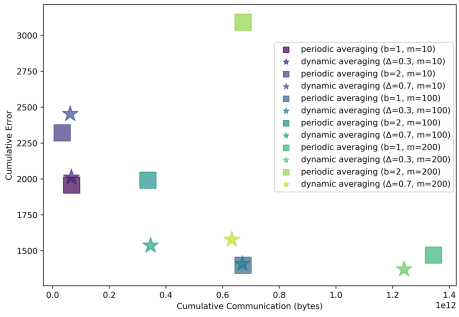
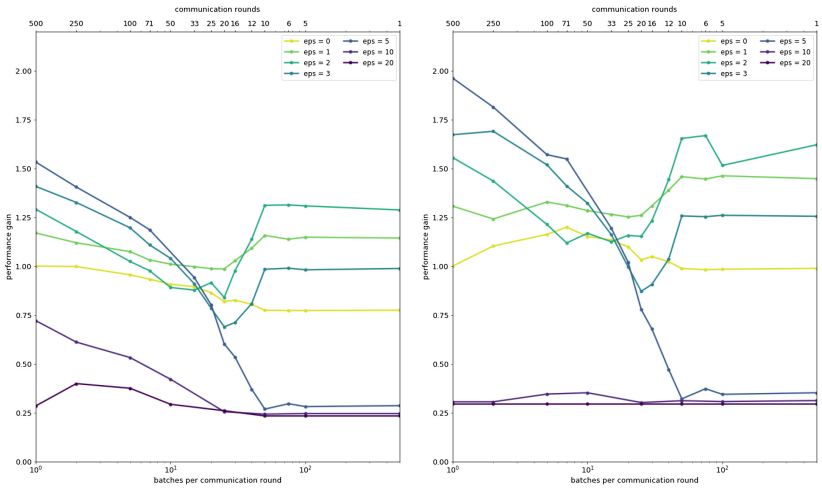


Fig. 7. Cumulative loss and cumulative communication of learning protocols for a different amount of learners. Training is performed on MNIST for 2, 20 and 40 epochs for $m = 10$, $m = 100$, $m = 200$ setups correspondingly.

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 [11,22]. 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 Fig. 7 and Appendix A.6 in the supplementary material for details).

A general question when using averaging is how local models should be initialized. McMahan et al. [22] 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 aver-



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

Fig. 8. Relative performances of averaged models on MNIST obtained from various heterogeneous model initializations parameterized by ϵ and various $b \in \mathbb{N}$. All averaged model performances are compared to an experiment with homogeneous model initializations ($\epsilon = 0$) and $b = 1$.

aging different initializations can indeed be beneficial. Figure 8 shows the performances of dynamic and periodic averaging for different numbers of rounds of averaging and different levels of inhomogeneity in the initializations. The results confirm that for one round of averaging, strongly inhomogeneous initializations deteriorate the learning process, but for more frequent rounds of averaging 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 periodically 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.

Acknowledgements. This research has been supported by the Center of Competence Machine Learning Rhein-Ruhr (ML2R).

References

1. Bojarski, M., et al.: End to end learning for self-driving cars. CoRR abs/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. Proc. Neuro-Nımes **91**(8), 12 (1991)
4. Bshouty, N.H., Long, P.M.: Linear classifiers are nearly optimal when hidden variables have diverse effects. Mach. Learn. **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. Dean, J., et al.: Large scale distributed deep networks. In: Advances in Neural Information Processing Systems, pp. 1223–1231 (2012)
7. Dekel, O., Gilad-Bachrach, R., Shamir, O., Xiao, L.: Optimal distributed online prediction using mini-batches. J. Mach. Learn. Res. **13**, 165–202 (2012)
8. Feng, J., Xu, H., Mannor, S.: Outlier robust online learning. CoRR abs/1701.00251 (2017)
9. 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)
10. 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, pp. 37–47. IEEE (2014)
11. 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)
12. Kamp, M., Boley, M., Keren, D., Schuster, A., Sharfman, I.: Communication-efficient distributed online prediction by dynamic model synchronization. In: Calders, T., Esposito, F., Hüllermeier, E., Meo, R. (eds.) ECML PKDD 2014. LNCS (LNAI), vol. 8724, pp. 623–639. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44848-9_40
13. Kamp, M., Boley, M., Missura, O., Gärtner, T.: Effective parallelisation for machine learning. In: Advances in Neural Information Processing Systems, pp. 6480–6491 (2017)
14. 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)
15. Kamp, M., Bothe, S., Boley, M., Mock, M.: Communication-efficient distributed online learning with kernels. In: Frasconi, P., Landwehr, N., Manco, G., Vreeken, J. (eds.) ECML PKDD 2016. LNCS (LNAI), vol. 9852, pp. 805–819. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46227-1_50

16. Keren, D., Sagy, G., Abboud, A., Ben-David, D., Schuster, A., Sharfman, I., Deligiannakis, A.: Geometric monitoring of heterogeneous streams. IEEE Trans. Knowl. Data Eng. **26**(8), 1890–1903 (2014)
17. Keren, D., Sharfman, I., Schuster, A., Livne, A.: Shape sensitive geometric monitoring. IEEE Trans. Knowl. Data Eng. **24**(8), 1520–1535 (2012)
18. 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)
19. Lazerson, A., Sharfman, I., Keren, D., Schuster, A., Garofalakis, M., Samoladas, V.: Monitoring distributed streams using convex decompositions. Proc. VLDB Endow. **8**(5), 545–556 (2015)
20. LeCun, Y.: The mnist database of handwritten digits (1998). http://yann.lecun.com/exdb/mnist/
21. 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)
22. 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)
23. McMahan, B., Ramage, D., Talwar, K., Zhang, L.: Learning differentially private recurrent language models. In: International Conference on Learning Representations (2018)
24. Moler, C.: Matrix computation on distributed memory multiprocessors. Hypercube Multiprocessors **86**(181–195), 31 (1986)
25. Nguyen, Q., Hein, M.: The loss surface of deep and wide neural networks. In: International Conference on Machine Learning, pp. 2603–2612 (2017)
26. Pomerleau, D.A.: Alvinn: an autonomous land vehicle in a neural network. In: Advances in Neural Information Processing Systems, pp. 305–313 (1989)
27. Shamir, O.: Without-replacement sampling for stochastic gradient methods. In: Advances in Neural Information Processing Systems, pp. 46–54 (2016)
28. 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)
29. Sharfman, I., Schuster, A., Keren, D.: A geometric approach to monitoring threshold functions over distributed data streams. Trans. Database Syst. **32**(4), 301–312 (2007)
30. 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)
31. 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)
32. 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)
33. Zhang, S., Choromanska, A.E., LeCun, Y.: Deep learning with elastic averaging sgd. In: Advances in Neural Information Processing Systems, pp. 685–693 (2015)

34. 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)
35. Zinkevich, M., Weimer, M., Smola, A.J., Li, L.: Parallelized stochastic gradient descent. In: Advances in Neural Information Processing Systems, pp. 2595–2603 (2010)