* Communication-Efﬁcient Learning of Deep Networksfrom Decentralized Data
* 7102
* b
* e
* F
* 82
* ]
* G
* L
* .
* s
* c
* [
* 3v92650
* .
* 2061
* :
* v
* i
* X
* r
* a
* H. Brendan McMahan
* Eider Moore
* Daniel Ramage
* Seth Hampson
* Google, Inc., 651 N 34th St., Seattle, WA 98103 USA
* Blaise Ag ¨uera y Arcas
* Abstract
* Modern mobile devices have access to a wealthof data suitable for learning models, which in turncan greatly improve the user experience on thedevice. For example, language models can im-prove speech recognition and text entry, and im-age models can automatically select good photos.However, this rich data is often privacy sensitive,large in quantity, or both, which may precludelogging to the data center and training there usingconventional approaches. We advocate an alter-native that leaves the training data distributed onthe mobile devices, and learns a shared model byaggregating locally-computed updates. We termthis decentralized approach Federated Learning.We present a practical method for the federatedlearning of deep networks based on iterativemodel averaging, and conduct an extensive empiri-cal evaluation, considering ﬁve different model ar-chitectures and four datasets. These experimentsdemonstrate the approach is robust to the unbal-anced and non-IID data distributions that are adeﬁning characteristic of this setting. Commu-nication costs are the principal constraint, andwe show a reduction in required communicationrounds by 10–100× as compared to synchronizedstochastic gradient descent.
* 1
* Introduction
* Increasingly, phones and tablets are the primary computingdevices for many people [30, 2]. The powerful sensors onthese devices (including cameras, microphones, and GPS),combined with the fact they are frequently carried, meansthey have access to an unprecedented amount of data, muchof it private in nature. Models learned on such data hold the
* Proceedings of the 20th International Conference on Artiﬁcial In-telligence and Statistics (AISTATS) 2017, Fort Lauderdale, Florida,USA. JMLR: W&CP volume 54. Copyright 2017 by the author(s).
* promise of greatly improving usability by powering moreintelligent applications, but the sensitive nature of the datameans there are risks and responsibilities to storing it in acentralized location.We investigate a learning technique that allows users tocollectively reap the beneﬁts of shared models trained fromthis rich data, without the need to centrally store it. We termour approach Federated Learning, since the learning task issolved by a loose federation of participating devices (whichwe refer to as clients) which are coordinated by a centralserver. Each client has a local training dataset which isnever uploaded to the server. Instead, each client computesan update to the current global model maintained by theserver, and only this update is communicated. This is adirect application of the principle of focused collection ordata minimization proposed by the 2012 White House reporton privacy of consumer data [39]. Since these updates arespeciﬁc to improving the current model, there is no reasonto store them once they have been applied.A principal advantage of this approach is the decoupling ofmodel training from the need for direct access to the rawtraining data. Clearly, some trust of the server coordinat-ing the training is still required. However, for applicationswhere the training objective can be speciﬁed on the basisof data available on each client, federated learning can sig-niﬁcantly reduce privacy and security risks by limiting theattack surface to only the device, rather than the device andthe cloud.Our primary contributions are 1) the identiﬁcation of theproblem of training on decentralized data from mobile de-vices as an important research direction; 2) the selection ofa straightforward and practical algorithm that can be appliedto this setting; and 3) an extensive empirical evaluation ofthe proposed approach. More concretely, we introduce theFederatedAveraging algorithm, which combines lo-cal stochastic gradient descent (SGD) on each client witha server that performs model averaging. We perform ex-tensive experiments on this algorithm, demonstrating it isrobust to unbalanced and non-IID data distributions, andcan reduce the rounds of communication needed to train adeep network on decentralized data by orders of magnitude.





* Communication-Efﬁcient Learning of Deep Networks from Decentralized Data
* Federated Learning Ideal problems for federated learn-ing have the following properties: 1) Training on real-worlddata from mobile devices provides a distinct advantage overtraining on proxy data that is generally available in the datacenter. 2) This data is privacy sensitive or large in size (com-pared to the size of the model), so it is preferable not to logit to the data center purely for the purpose of model training(in service of the focused collection principle). 3) For super-vised tasks, labels on the data can be inferred naturally fromuser interaction.Many models that power intelligent behavior on mobiledevices ﬁt the above criteria. As two examples, we con-sider image classiﬁcation, for example predicting whichphotos are most likely to be viewed multiple times in thefuture, or shared; and language models, which can be usedto improve voice recognition and text entry on touch-screenkeyboards by improving decoding, next-word-prediction,and even predicting whole replies [10]. The potential train-ing data for both these tasks (all the photos a user takes andeverything they type on their mobile keyboard, includingpasswords, URLs, messages, etc.) can be privacy sensitive.The distributions from which these examples are drawn arealso likely to differ substantially from easily available proxydatasets: the use of language in chat and text messages isgenerally much different than standard language corpora,e.g., Wikipedia and other web documents; the photos peopletake on their phone are likely quite different than typicalFlickr photos. And ﬁnally, the labels for these problems aredirectly available: entered text is self-labeled for learninga language model, and photo labels can be deﬁned by natu-ral user interaction with their photo app (which photos aredeleted, shared, or viewed).Both of these tasks are well-suited to learning a neural net-work. For image classiﬁcation feed-forward deep networks,and in particular convolutional networks, are well-knownto provide state-of-the-art results [26, 25]. For languagemodeling tasks recurrent neural networks, and in particularLSTMs, have achieved state-of-the-art results [20, 5, 22].
* Privacy Federated learning has distinct privacy advan-tages compared to data center training on persisted data.Holding even an “anonymized” dataset can still put userprivacy at risk via joins with other data [37]. In contrast,the information transmitted for federated learning is theminimal update necessary to improve a particular model(naturally, the strength of the privacy beneﬁt depends on thecontent of the updates).1 The updates themselves can (andshould) be ephemeral. They will never contain more infor-
* 1 For example, if the update is the total gradient of the loss onall of the local data, and the features are a sparse bag-of-words,then the non-zero gradients reveal exactly which words the userhas entered on the device. In contrast, the sum of many gradientsfor a dense model such as a CNN offers a harder target for attackersseeking information about individual training instances (thoughattacks are still possible).
* mation than the raw training data (by the data processinginequality), and will generally contain much less. Further,the source of the updates is not needed by the aggregationalgorithm, so updates can be transmitted without identifyingmeta-data over a mix network such as Tor [7] or via a trustedthird party. We brieﬂy discuss the possibility of combiningfederated learning with secure multiparty computation anddifferential privacy at the end of the paper.
* Federated Optimization We refer to the optimizationproblem implicit in federated learning as federated optimiza-tion, drawing a connection (and contrast) to distributed opti-mization. Federated optimization has several key propertiesthat differentiate it from a typical distributed optimizationproblem:• Non-IID The training data on a given client is typicallybased on the usage of the mobile device by a particularuser, and hence any particular user’s local dataset willnot be representative of the population distribution.• Unbalanced Similarly, some users will make muchheavier use of the service or app than others, leadingto varying amounts of local training data.
* • Massively distributed We expect the number of
* clients participating in an optimization to be muchlarger than the average number of examples per client.
* • Limited communication Mobile devices are fre-
* quently ofﬂine or on slow or expensive connections.In this work, our emphasis is on the non-IID and unbalancedproperties of the optimization, as well as the critical natureof the communication constraints. A deployed federatedoptimization system must also address a myriad of practicalissues: client datasets that change as data is added anddeleted; client availability that correlates with the local datadistribution in complex ways (e.g., phones from speakersof American English will likely be plugged in at differenttimes than speakers of British English); and clients thatnever respond or send corrupted updates.These issues are beyond the scope of the current work;instead, we use a controlled environment that is suitablefor experiments, but still addresses the key issues of clientavailability and unbalanced and non-IID data. We assumea synchronous update scheme that proceeds in rounds ofcommunication. There is a ﬁxed set of K clients, eachwith a ﬁxed local dataset. At the beginning of each round,a random fraction C of clients is selected, and the serversends the current global algorithm state to each of theseclients (e.g., the current model parameters). We only selecta fraction of clients for efﬁciency, as our experiments showdiminishing returns for adding more clients beyond a certainpoint. Each selected client then performs local computationbased on the global state and its local dataset, and sends anupdate to the server. The server then applies these updatesto its global state, and the process repeats.
* H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, Blaise Ag ¨uera y Arcas
* While we focus on non-convex neural network objectives,the algorithm we consider is applicable to any ﬁnite-sumobjective of the form
* min
* w∈Rd
* f (w)
* where
* f (w)
* def
* =
* 1n
* fi (w).
* (1)
* n(cid:88)
* i=1
* K(cid:88)
* nkn
* (cid:88)
* 1nk
* For a machine learning problem, we typically take fi (w) =(cid:96)(xi , yi ; w), that is, the loss of the prediction on example(xi , yi ) made with model parameters w . We assume thereare K clients over which the data is partitioned, with Pk theset of indexes of data points on client k , with nk = |Pk |.Thus, we can re-write the objective (1) as
* fi (w).
* f (w) =
* Fk (w) where Fk (w) =
* k=1
* i∈PkIf the partition Pk was formed by distributing the trainingexamples over the clients uniformly at random, then wewould have EPk [Fk (w)] = f (w), where the expectation isover the set of examples assigned to a ﬁxed client k . This isthe IID assumption typically made by distributed optimiza-tion algorithms; we refer to the case where this does nothold (that is, Fk could be an arbitrarily bad approximationto f ) as the non-IID setting.In data center optimization, communication costs are rela-tively small, and computational costs dominate, with muchof the recent emphasis being on using GPUs to lower thesecosts. In contrast, in federated optimization communicationcosts dominate — we will typically be limited by an uploadbandwidth of 1 MB/s or less. Further, clients will typicallyonly volunteer to participate in the optimization when theyare charged, plugged-in, and on an unmetered wi-ﬁ connec-tion. Further, we expect each client will only participate in asmall number of update rounds per day. On the other hand,since any single on-device dataset is small compared to thetotal dataset size, and modern smartphones have relativelyfast processors (including GPUs), computation becomesessentially free compared to communication costs for manymodel types. Thus, our goal is to use additional computationin order to decrease the number of rounds of communica-tion needed to train a model. There are two primary wayswe can add computation: 1) increased parallelism, wherewe use more clients working independently between eachcommunication round; and, 2) increased computation oneach client, where rather than performing a simple computa-tion like a gradient calculation, each client performs a morecomplex calculation between each communication round.We investigate both of these approaches, but the speedupswe achieve are due primarily to adding more computationon each client, once a minimum level of parallelism overclients is used.
* Related Work Distributed training by iteratively averag-ing locally trained models has been studied by McDon-ald et al. [28] for the perceptron and Povey et al. [31] for
* speech recognition DNNs. Zhang et al. [42] studies an asyn-chronous approach with “soft” averaging. These works onlyconsider the cluster / data center setting (at most 16 workers,wall-clock time based on fast networks), and do not considerdatasets that are unbalanced and non-IID, properties thatare essential to the federated learning setting. We adaptthis style of algorithm to the federated setting and performthe appropriate empirical evaluation, which asks differentquestions than those relevant in the data center setting, andrequires different methodology.Using similar motivation to ours, Neverova et al. [29] alsodiscusses the advantages of keeping sensitive user data ondevice. The work of Shokri and Shmatikov [35] is related inseveral ways: they focus on training deep networks, empha-size the importance of privacy, and address communicationcosts by only sharing a subset of the parameters during eachround of communication; however, they also do not considerunbalanced and non-IID data, and the empirical evaluationis limited.In the convex setting, the problem of distributed opti-mization and estimation has received signiﬁcant attention[4, 15, 33], and some algorithms do focus speciﬁcally oncommunication efﬁciency [45, 34, 40, 27, 43]. In additionto assuming convexity, this existing work generally requiresthat the number of clients is much smaller than the numberof examples per client, that the data is distributed acrossthe clients in IID fashion, and that each node has an iden-tical number of data points — all of these assumptionsare violated in the federated optimization setting. Asyn-chronous distributed forms of SGD have also been appliedto training neural networks, e.g., Dean et al. [12], but theseapproaches require a prohibitive number of updates in thefederated setting. Distributed consensus algorithms (e.g.,[41]) relax the IID assumption, but are still not a good ﬁt forcommunication-constrained optimization over very manyclients.One endpoint of the (parameterized) algorithm family weconsider is simple one-shot averaging, where each clientsolves for the model that minimizes (possibly regularized)loss on their local data, and these models are averaged toproduce the ﬁnal global model. This approach has beenstudied extensively in the convex case with IID data, and itis known that in the worst-case, the global model produced isno better than training a model on a single client [44, 3, 46].
* 2 The FederatedAveraging Algorithm
* The recent multitude of successful applications of deeplearning have almost exclusively relied on variants ofstochastic gradient descent (SGD) for optimization; in fact,many advances can be understood as adapting the struc-ture of the model (and hence the loss function) to be moreamenable to optimization by simple gradient-based meth-ods [16]. Thus, it is natural that we build algorithms for
* Communication-Efﬁcient Learning of Deep Networks from Decentralized Data
* federated optimization by starting from SGD.SGD can be applied naively to the federated optimizationproblem, where a single batch gradient calculation (say ona randomly selected client) is done per round of commu-nication. This approach is computationally efﬁcient, butrequires very large numbers of rounds of training to producegood models (e.g., even using an advanced approach likebatch normalization, Ioffe and Szegedy [21] trained MNISTfor 50000 steps on minibatches of size 60). We considerthis baseline in our CIFAR-10 experiments.In the federated setting, there is little cost in wall-clock timeto involving more clients, and so for our baseline we uselarge-batch synchronous SGD; experiments by Chen et al.[8] show this approach is state-of-the-art in the data centersetting, where it outperforms asynchronous approaches. Toapply this approach in the federated setting, we select a C -fraction of clients on each round, and compute the gradientof the loss over all the data held by these clients. Thus, Ccontrols the global batch size, with C = 1 correspondingto full-batch (non-stochastic) gradient descent.2 We refer tothis baseline algorithm as FederatedSGD (or FedSGD).A typical implementation of FedSGD with C = 1 anda ﬁxed learning rate η has each client k compute gk =and applies the update wt+1 ← wt − η (cid:80)K(cid:79)Fk (wt ), the average gradient on its local data at the currentmodel wt , and the central server aggregates these gradientsn gk , sincen gk = (cid:79)f (wt ). An equivalent update is given by
* t+1 ← wt − ηgk and then wt+1 ← (cid:80)K∀k , wk
* (cid:80)K
* nkn wkt+1 .
* k=1
* That is, each client locally takes one step of gradient de-scent on the current model using its local data, and theserver then takes a weighted average of the resulting models.Once the algorithm is written this way, we can add morecomputation to each client by iterating the local updatewk ← wk − η(cid:79)Fk (wk ) multiple times before the averag-ing step. We term this approach FederatedAveraging(or FedAvg). The amount of computation is controlledby three key parameters: C , the fraction of clients thatperform computation on each round; E , then number oftraining passes each client makes over its local dataset oneach round; and B , the local minibatch size used for theclient updates. We write B = ∞ to indicate that the fulllocal dataset is treated as a single minibatch. Thus, at oneendpoint of this algorithm family, we can take B = ∞ andE = 1 which corresponds exactly to FedSGD. For a clientwith nk local examples, the number of local updates perround is given by uk = E nkB ; Complete pseudo-code isgiven in Algorithm 1.For general non-convex objectives, averaging models inparameter space could produce an arbitrarily bad model.
* nk
* k=1
* nk
* k=1
* Figure 1: The loss on the full MNIST training set for modelsgenerated by averaging the parameters of two models wand w (cid:48) using θw + (1 − θ)w (cid:48) for 50 evenly spaced valuesθ ∈ [−0.2, 1.2].The models w and w (cid:48) were trained usingSGD on different small datasets. For the left plot, w and w (cid:48)were initialized using different random seeds; for the rightplot, a shared seed was used. Note the different y -axis scales.The horizontal line gives the best loss achieved by w or w (cid:48)(which were quite close, corresponding to the vertical linesat θ = 0 and θ = 1). With shared initialization, averagingthe models produces a signiﬁcant reduction in the loss onthe total training set (much better than the loss of eitherparent model).
* Following the approach of Goodfellow et al. [17], we seeexactly this bad behavior when we average two MNISTdigit-recognition models3 trained from different initial con-ditions (Figure 1, left). For this ﬁgure, the parent models wand w (cid:48) were each trained on non-overlapping IID samplesof 600 examples from the MNIST training set. Trainingwas via SGD with a ﬁxed learning rate of 0.1 for 240 up-dates on minibatches of size 50 (or E = 20 passes overthe mini-datasets of size 600). This is approximately theamount of training where the models begin to overﬁt theirlocal datasets.Recent work indicates that in practice, the loss surfaces ofsufﬁciently over-parameterized NNs are surprisingly well-behaved and in particular less prone to bad local minimathan previously thought [11, 17, 9]. And indeed, when westart two models from the same random initialization andthen again train each independently on a different subset ofthe data (as described above), we ﬁnd that naive parameteraveraging works surprisingly well (Figure 1, right): the av-erage of these two models, 12 w (cid:48) , achieves signiﬁcantlylower loss on the full MNIST training set than the bestmodel achieved by training on either of the small datasetsindependently. While Figure 1 starts from a random initial-ization, note a shared starting model wt is used for eachround of FedAvg, and so the same intuition applies.
* 2 w + 1
* 2 While the batch selection mechanism is different than se-lecting a batch by choosing individual examples uniformly atrandom, the batch gradients g computed by FedSGD still satisfy
* E[g ] = (cid:79)f (w).
* 3We use the “2NN” multi-layer perceptron described in Sec-tion 3.
* H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, Blaise Ag ¨uera y Arcas
* Algorithm 1 FederatedAveraging. The K clients areindexed by k ; B is the local minibatch size, E is the numberof local epochs, and η is the learning rate.
* Server executes:
* m ← max(C · K, 1)
* initialize w0for each round t = 1, 2, . . . doSt ← (random set of m clients)t+1 ← ClientUpdate(k , wt )
* for each client k ∈ St in parallel do
* wt+1 ← (cid:80)Kk=1
* wk
* nkn wkt+1
* ClientUpdate(k , w):
* // Run on client kB ← (split Pk into batches of size B )for each local epoch i from 1 to E do
* for batch b ∈ B do
* w ← w − η(cid:79)(cid:96)(w; b)
* return w to server
* 3 Experimental Results
* We are motivated by both image classiﬁcation and languagemodeling tasks where good models can greatly enhance theusability of mobile devices. For each of these tasks we ﬁrstpicked a proxy dataset of modest enough size that we couldthoroughly investigate the hyperparameters of the FedAvgalgorithm. While each individual training run is relativelysmall, we trained over 2000 individual models for theseexperiments. We then present results on the benchmarkCIFAR-10 image classiﬁcation task. Finally, to demonstratethe effectiveness of FedAvg on a real-world problem witha natural partitioning of the data over clients, we evaluateon a large language modeling task.Our initial study includes three model families on twodatasets. The ﬁrst two are for the MNIST digit recognitiontask [26]: 1) A simple multilayer-perceptron with 2-hiddenlayers with 200 units each using ReLu activations (199,210total parameters), which we refer to as the MNIST 2NN.2) A CNN with two 5x5 convolution layers (the ﬁrst with32 channels, the second with 64, each followed with 2x2max pooling), a fully connected layer with 512 units andReLu activation, and a ﬁnal softmax output layer (1,663,370total parameters). To study federated optimization, we alsoneed to specify how the data is distributed over the clients.We study two ways of partitioning the MNIST data overclients: IID, where the data is shufﬂed, and then partitionedinto 100 clients each receiving 600 examples, and Non-IID,where we ﬁrst sort the data by digit label, divide it into 200shards of size 300, and assign each of 100 clients 2 shards.This is a pathological non-IID partition of the data, as mostclients will only have examples of two digits. Thus, this letsus explore the degree to which our algorithms will break onhighly non-IID data. Both of these partitions are balanced,
* Table 1: Effect of the client fraction C on the MNIST 2NNwith E = 1 and CNN with E = 5. Note C = 0.0 corre-sponds to one client per round; since we use 100 clients forthe MNIST data, the rows correspond to 1, 10 20, 50, and100 clients. Each table entry gives the number of roundsof communication necessary to achieve a test-set accuracyof 97% for the 2NN and 99% for the CNN, along with thespeedup relative to the C = 0 baseline. Five runs withthe large batch size did not reach the target accuracy in theallowed time.
* 2NN
* C
* I ID
* B = ∞
* 0 .0 14550 .1 1474 (1.0×)0 .2 1658 (0.9×)0 .5 — (— )1 .0 — (— )CNN , E = 50 .03870 .10 .20 .51 .0
* 339 (1.1×)337 (1.1×)164 (2.4×)246 (1.6×)
* 316
* B = 1087 (3.6×)77 (4.1×)75 (4.2×)70 (4.5×)
* 50
* 18 (2.8×)18 (2.8×)18 (2.8×)16 (3.1×)
* NON - I ID
* B = ∞
* 42781796 (2.4×)1528 (2.8×)— (— )— (— )
* 978 (1.2×)
* 11811100 (1.1×)1067 (1.1×)— (— )
* 3275
* B = 10664 (4.9×)619 (5.3×)443 (7.4×)380 (8.6×)
* 956
* 206 (4.6×)200 (4.8×)261 (3.7×)97 (9.9×)
* however.4For language modeling, we built a dataset from The Com-plete Works of William Shakespeare [32]. We construct aclient dataset for each speaking role in each play with atleast two lines. This produced a dataset with 1146 clients.For each client, we split the data into a set of training lines(the ﬁrst 80% of lines for the role), and test lines (the last20%, rounded up to at least one line). The resulting datasethas 3,564,579 characters in the training set, and 870,014characters5 in the test set. This data is substantially unbal-anced, with many roles having only a few lines, and a fewwith a large number of lines. Further, observe the test set isnot a random sample of lines, but is temporally separatedby the chronology of each play. Using an identical train/testsplit, we also form a balanced and IID version of the dataset,also with 1146 clients.On this data we train a stacked character-level LSTM lan-guage model, which after reading each character in a line,predicts the next character [22]. The model takes a series ofcharacters as input and embeds each of these into a learned8 dimensional space. The embedded characters are thenprocessed through 2 LSTM layers, each with 256 nodes.Finally the output of the second LSTM layer is sent to asoftmax output layer with one node per character. The fullmodel has 866,578 parameters, and we trained using anunroll length of 80 characters.
* 4We performed additional experiments on unbalanced versionsof these datasets, and found them to in fact be slightly easier for
* FedAvg.
* 5We always use character to refer to a one byte string, and userole to refer to a part in the play.
* Communication-Efﬁcient Learning of Deep Networks from Decentralized Data
* Table 2: Number of communication rounds to reach a targetaccuracy for FedAvg, versus FedSGD (ﬁrst row, E = 1and B = ∞). The u column gives u = En/(KB ), theexpected number of updates per round.
* CNN
* F E DSGD
* F E DA V GF E DA V GF E DA V GF E DA V GF E DA V GF E DA V GF E DA V GF E DA V G
* LSTM
* F E DSGD
* F E DA V GF E DA V GF E DA V GF E DA V GF E DA V G
* MN IST CNN , 99% ACCURACYI ID
* B
* E
* u
* 1 ∞5 ∞
* 15020 ∞11055020505102010
* 15122060602403001200
* 626
* 179 (3.5×)65 (9.6×)234 (2.7×)34 (18.4×)29 (21.6×)32 (19.6×)20 (31.3×)18 (34.8×)
* SHAKE S PEARE LSTM , 54% ACCURACY
* E
* 155
* B
* 1 ∞5 ∞
* 50
* 1
* 105010
* u
* 1 .01 .55 .07 .47 .437 .1
* I ID24881635 (1.5×)
* 613 (4.1×)460 (5.4×)401 (6.2×)192 (13.0×)
* NON - I ID4831000 (0.5×)
* 600 (0.8×)672 (0.7×)350 (1.4×)334 (1.4×)426 (1.1×)229 (2.1×)173 (2.8×)
* NON - I ID3906
* 549 (7.1×)597 (6.5×)164 (23.8×)152 (25.7×)41 (95.3×)
* 10 1
* SGD is sensitive to the tuning of the learning-rate parameterη . The results reported here are based on training overa sufﬁciently wide grid of learning rates (typically 11-13values for η on a multiplicative grid of resolution 10 13 or6 ). We checked to ensure the best learning rates were inthe middle of our grids, and that there was not a signiﬁcantdifference between the best learning rates. Unless otherwisenoted, we plot metrics for the best performing rate selectedindividually for each x-axis value. We ﬁnd that the optimallearning rates do not vary too much as a function of theother parameters.
* Increasing parallelism We ﬁrst experiment with theclient fraction C , which controls the amount of multi-clientparallelism. Table 1 shows the impact of varying C for bothMNIST models. We report the number of communicationrounds necessary to achieve a target test-set accuracy. Tocompute this, we construct a learning curve for each com-bination of parameter settings, optimizing η as describedabove and then making each curve monotonically improvingby taking the best value of test-set accuracy achieved overall prior rounds. We then calculate the number of roundswhere the curve crosses the target accuracy, using linearinterpolation between the discrete points forming the curve.This is perhaps best understood by reference to Figure 2,where the gray lines show the targets.With B = ∞ (for MNIST processing all 600 client ex-amples as a single batch per round), there is only a smalladvantage in increasing the client fraction. Using the smallerbatch size B = 10 shows a signiﬁcant improvement in usingC ≥ 0.1, especially in the non-IID case. Based on theseresults, for most of the remainder of our experiments weﬁx C = 0.1, which strikes a good balance between com-putational efﬁciency and convergence rate. Comparing thenumber of rounds for the B = ∞ and B = 10 columns in
* Figure 2: Test set accuracy vs. communication roundsfor the MNIST CNN (IID and then pathological non-IID)and Shakespeare LSTM (IID and then by Play&Role) withC = 0.1 and optimized η . The gray lines show the targetaccuracies used in Table 2. Plots for the 2NN are given asFigure 7 in Appendix A.
* Table 1 shows a dramatic speedup, which we investigatenext.
* Increasing computation per client
* In this section, weﬁx C = 0.1, and add more computation per client on eachround, either decreasing B , increasing E , or both. Figure 2demonstrates that adding more local SGD updates per roundcan produce a dramatic decrease in communication costs,and Table 2 quantiﬁes these speedups. The expected numberof updates per client per round is u = (E[nk ]/B )E =nE /(KB ), where the expectation is over the draw of arandom client k . We order the rows in each section ofTable 2 by this statistic. We see that increasing u by varyingboth E and B is effective. As long as B is large enoughto take full advantage of available parallelism on the clienthardware, there is essentially no cost in computation timefor lowering it, and so in practice this should be the ﬁrstparameter tuned.For the IID partition of the MNIST data, using more compu-tation per client decreases the number of rounds to reach thetarget accuracy by 35× for the CNN and 46× for the 2NN(see Table 4 in Appendix A for details for the 2NN). Thespeedups for the pathologically partitioned non-IID data aresmaller, but still substantial (2.8 – 3.7×). It is impressive
* H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, Blaise Ag ¨uera y Arcas
* Figure 3: The effect of training for many local epochs (largeE ) between averaging steps, ﬁxing B = 10 and C = 0.1 forthe Shakespeare LSTM with a ﬁxed learning rate η = 1.47.
* Figure 4: Test accuracy versus communication for the CI-FAR10 experiments. FedSGD uses a learning-rate decayof 0.9934 per round; FedAvg uses B = 50, learning-ratedecay of 0.99 per round, and E = 5.
* that averaging provides any advantage (vs. actually diverg-ing) when we naively average the parameters of modelstrained on entirely different pairs of digits. Thus, we viewthis as strong evidence for the robustness of this approach.The unbalanced and non-IID distribution of the Shakespeare(by role in the play) is much more representative of the kindof data distribution we expect for real-world applications.Encouragingly, for this problem learning on the non-IID andunbalanced data is actually much easier (a 95× speedup vs13× for the balanced IID data); we conjecture this is largelydue to the fact some roles have relatively large local datasets,which makes increased local training particularly valuable.For all three model classes, FedAvg converges to a higherlevel of test-set accuracy than the baseline FedSGD mod-els. This trend continues even if the lines are extendedbeyond the plotted ranges. For example, for the CNN theB = ∞, E = 1 FedSGD model eventually reaches 99.22%accuracy after 1200 rounds (and had not improved furtherafter 6000 rounds), while the B = 10, E = 20 FedAvgmodel reaches an accuracy of 99.44% after 300 rounds. Weconjecture that in addition to lowering communication costs,model averaging produces a regularization beneﬁt similarto that achieved by dropout [36].We are primarily concerned with generalization perfor-mance, but FedAvg is effective at optimizing the trainingloss as well, even beyond the point where test-set accuracyplateaus. We observed similar behavior for all three modelclasses, and present plots for the MNIST CNN in Figure 6in Appendix A.
* Can we over-optimize on the client datasets? The cur-
* rent model parameters only inﬂuence the optimization per-formed in each ClientUpdate via initialization. Thus,as E → ∞, at least for a convex problem eventually theinitial conditions should be irrelevant, and the global min-imum would be reached regardless of initialization. Evenfor a non-convex problem, one might conjecture the algo-
* rithm would converge to the same local minimum as long asthe initialization was in the same basin. That is, we wouldexpect that while one round of averaging might produce areasonable model, additional rounds of communication (andaveraging) would not produce further improvements.Figure 3 shows the impact of large E during initial trainingon the Shakespeare LSTM problem. Indeed, for very largenumbers of local epochs, FedAvg can plateau or diverge.6This result suggests that for some models, especially in thelater stages of convergence, it may be useful to decay theamount of local computation per round (moving to smallerE or larger B ) in the same way decaying learning ratescan be useful. Figure 8 in Appendix A gives the analo-gous experiment for the MNIST CNN. Interestingly, for thismodel we see no signiﬁcant degradation in the convergencerate for large values of E . However, we see slightly betterperformance for E = 1 versus E = 5 for the large-scalelanguage modeling task described below (see Figure 10 inAppendix A).
* CIFAR experiments We also ran experiments on theCIFAR-10 dataset [24] to further validate FedAvg. Thedataset consists of 10 classes of 32x32 images with threeRGB channels. There are 50,000 training examples and10,000 testing examples, which we partitioned into 100clients each containing 500 training and 100 testing exam-ples; since there isn’t a natural user partitioning of this data,we considered the balanced and IID setting. The modelarchitecture was taken from the TensorFlow tutorial [38],which consists of two convolutional layers followed by twofully connected layers and then a linear transformation layerto produce logits, for a total of about 106 parameters. Note
* 6 Note that due to this behavior and because for large E not allexperiments for all learning rates were run for the full number ofrounds, we report results for a ﬁxed learning rate (which perhapssurprisingly was near-optimal across the range of E parameters)and without forcing the lines to be monotonic.
* Communication-Efﬁcient Learning of Deep Networks from Decentralized Data
* Table 3: Number of rounds and speedup relative to baselineSGD to reach a target test-set accuracy on CIFAR10. SGDused a minibatch size of 100. FedSGD and FedAvg used
* C = 0.1, with FedAvg using E = 5 and B = 50.
* ACC .
* SGDF E DSGD
* F E DA V G
* 80%18000(— )3750 (4 .8×)280 (64 .3×)
* 82%31000(— )6600 (4 .7×)630 (49 .2×)
* 85%99000(— )2000 (49 .5×)(— )
* N /A
* that state-of-the-art approaches have achieved a test accu-racy of 96.5% [19] for CIFAR; nevertheless, the standardmodel we use is sufﬁcient for our needs, as our goal is toevaluate our optimization method, not achieve the best pos-sible accuracy on this task. The images are preprocessed aspart of the training input pipeline, which consists of crop-ping the images to 24x24, randomly ﬂipping left-right andadjusting the contrast, brightness and whitening.For these experiments, we considered an additional base-line, standard SGD training on the full training set (no userpartitioning), using minibatches of size 100. We achievedan 86% test accuracy after 197,500 minibatch updates(each minibatch update requires a communication roundin the federated setting). FedAvg achieves a similar testaccuracy of 85% after only 2,000 communication rounds.For all algorithms, we tuned a learning-rate decay param-eter in addition to the initial learning rate. Table 3 givesthe number of communication rounds for baseline SGD,FedSGD, and FedAvg to reach three different accuracytargets, and Figure 4 gives learning-rate curves for FedAvgversus FedSGD.By running experiments with minibatches of size B = 50for both SGD and FedAvg, we can also look at accuracy asa function of the number of such minibatch gradient calcula-tions. We expect SGD to do better here, because a sequentialstep is taken after each minibatch computation. However,as Figure 9 in the appendix shows, for modest values of Cand E , FedAvg makes a similar amount of progress perminibatch computation. Further, we see that both standardSGD and FedAvg with only one client per round (C = 0),demonstrate signiﬁcant oscillations in accuracy, whereasaveraging over more clients smooths this out.
* Large-scale LSTM experiments We ran experiments on
* a large-scale next-word prediction task task to demonstratethe effectiveness of our approach on a real-world problem.Our training dataset consists 10 million public posts from alarge social network. We grouped the posts by author, fora total of over 500,000 clients. This dataset is a realisticproxy for the type of text entry data that would be presenton a user’s mobile device. We limited each client datasetto at most 5000 words, and report accuracy (the fractionof the data where the highest predicted probability was onthe correct next word, out of 10000 possibilities) on a testset of 1e5 posts from different (non-training) authors. Our
* Figure 5: Monotonic learning curves for the large-scalelanguage model word LSTM.
* model is a 256 node LSTM on a vocabulary of 10,000 words.The input and output embeddings for each word were ofdimension 192, and co-trained with the model; there are4,950,544 parameters in all. We used an unroll of 10 words.These experiments required signiﬁcant computational re-sources and so we did not explore hyper-parameters as thor-oughly: all runs trained on 200 clients per round; FedAvgused B = 8 and E = 1. We explored a variety of learn-ing rates for FedAvg and the baseline FedSGD. Figure 5shows monotonic learning curves for the best learning rates.FedSGD with η = 18.0 required 820 rounds to reach 10.5%accuracy, while FedAvg with η = 9.0 reached an accuracyof 10.5% in only 35 communication rounds (23× fewer thenFedSGD). We observed lower variance in test accuracy forFedAvg, see Figure 10 in Appendix A. This ﬁgure alsoinclude results for E = 5, which performed slightly worsethan E = 1.
* 4 Conclusions and Future Work
* Our experiments show that federated learning can be madepractical, as FedAvg trains high-quality models using rel-atively few rounds of communication, as demonstrated byresults on a variety of model architectures: a multi-layerperceptron, two different convolutional NNs, a two-layercharacter LSTM, and a large-scale word-level LSTM.While federated learning offers many practical privacy ben-eﬁts, providing stronger guarantees via differential pri-vacy [14, 13, 1], secure multi-party computation [18], ortheir combination is an interesting direction for future work.Note that both classes of techniques apply most naturally tosynchronous algorithms like FedAvg.7
* 7Subsequent to this work, Bonawitz et al. [6] introduced anefﬁcient secure aggregation protocol for federated learning, andKone ˇcn ´y et al. [23] presented algorithms for further decreasingcommunication costs.
* H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, Blaise Ag ¨uera y Arcas
* References
* [1] Martin Abadi, Andy Chu, Ian Goodfellow, BrendanMcMahan, Ilya Mironov, Kunal Talwar, and Li Zhang.Deep learning with differential privacy. In 23rd ACMConference on Computer and Communications Secu-rity (ACM CCS), 2016.[2] MonicaAnderson.viceownership:
* http://www.pewinternet.org/2015/10/29/technology-device-ownership-2015/,
* Technology
* 2015.
* de-
* 2015.[3] Yossi Arjevani and Ohad Shamir. Communicationcomplexity of distributed convex learning and opti-mization. In Advances in Neural Information Process-ing Systems 28. 2015.[4] Maria-Florina Balcan, Avrim Blum, Shai Fine, andYishay Mansour. Distributed learning, commu-nication complexity and privacy.arXiv preprintarXiv:1204.3514, 2012.[5] Yoshua Bengio, R ´ejean Ducharme, Pascal Vincent,and Christian Janvin. A neural probabilistic languagemodel. J. Mach. Learn. Res., 2003.[6] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Anto-nio Marcedone, H. Brendan McMahan, Sarvar Patel,Daniel Ramage, Aaron Segal, and Karn Seth. Practicalsecure aggregation for federated learning on user-helddata. In NIPS Workshop on Private Multi-Party Ma-chine Learning, 2016.[7] David L. Chaum. Untraceable electronic mail, returnaddresses, and digital pseudonyms. Commun. ACM,24(2), 1981.[8] Jianmin Chen, Rajat Monga, Samy Bengio, and RafalJozefowicz. Revisiting distributed synchronous sgd.In ICLR Workshop Track, 2016.[9] Anna Choromanska, Mikael Henaff, Micha ¨el Mathieu,G ´erard Ben Arous, and Yann LeCun. The loss surfacesof multilayer networks. In AISTATS, 2015.[10] GregCorrado.Computer,spondtothis
* http://googleresearch.blogspot.com/2015/11/computer-respond-to-this-email.
* email.
* re-
* html, November 2015.[11] Yann N. Dauphin, Razvan Pascanu, C¸ aglar G ¨ulc¸ ehre,KyungHyun Cho, Surya Ganguli, and Yoshua Bengio.Identifying and attacking the saddle point problem inhigh-dimensional non-convex optimization. In NIPS,2014.[12] Jeffrey Dean, Greg S. Corrado, Rajat Monga, KaiChen, Matthieu Devin, Quoc V. Le, Mark Z. Mao,Marc’Aurelio Ranzato, Andrew Senior, Paul Tucker,Ke Yang, and Andrew Y. Ng. Large scale distributeddeep networks. In NIPS, 2012.
* [13] John Duchi, Michael I. Jordan, and Martin J. Wain-wright. Privacy aware learning. Journal of the Associ-ation for Computing Machinery, 2014.[14] Cynthia Dwork and Aaron Roth. The AlgorithmicFoundations of Differential Privacy. Foundations andTrends in Theoretical Computer Science. Now Pub-lishers, 2014.[15] Olivier Fercoq, Zheng Qu, Peter Richt ´arik, and MartinTak ´ac. Fast distributed coordinate descent for non-strongly convex losses.In Machine Learning forSignal Processing (MLSP), 2014 IEEE InternationalWorkshop on, 2014.[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville.Deep learning. Book in preparation for MIT Press,2016.[17] Ian J. Goodfellow, Oriol Vinyals, and Andrew M. Saxe.Qualitatively characterizing neural network optimiza-tion problems. In ICLR, 2015.[18] Slawomir Goryczka, Li Xiong, and Vaidy Sunderam.Secure multiparty aggregation with differential pri-vacy: A comparative study.In Proceedings of theJoint EDBT/ICDT 2013 Workshops, 2013.[19] Benjamin Graham. Fractional max-pooling. CoRR,abs/1412.6071, 2014. URL http://arxiv.org/
* abs/1412.6071.
* [20] Sepp Hochreiter and J ¨urgen Schmidhuber. Long short-term memory. Neural Computation, 9(8), November1997.[21] Sergey Ioffe and Christian Szegedy. Batch normaliza-tion: Accelerating deep network training by reducinginternal covariate shift. In ICML, 2015.[22] Yoon Kim, Yacine Jernite, David Sontag, and Alexan-der M. Rush. Character-aware neural language models.CoRR, abs/1508.06615, 2015.[23] Jakub Kone ˇcn ´y, H. Brendan McMahan, Felix X. Yu,Peter Richtarik, Ananda Theertha Suresh, and DaveBacon. Federated learning: Strategies for improvingcommunication efﬁciency. In NIPS Workshop on Pri-vate Multi-Party Machine Learning, 2016.[24] Alex Krizhevsky. Learning multiple layers of featuresfrom tiny images. Technical report, 2009.[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hin-ton. Imagenet classiﬁcation with deep convolutionalneural networks. In NIPS. 2012.[26] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner.Gradient-based learning applied to document recogni-tion. Proceedings of the IEEE, 86(11), 1998.[27] Chenxin Ma, Virginia Smith, Martin Jaggi, Michael IJordan, Peter Richt ´arik, and Martin Tak ´a ˇc. Adding vs.averaging in distributed primal-dual optimization. InICML, 2015.
* Communication-Efﬁcient Learning of Deep Networks from Decentralized Data
* [43] Yuchen Zhang and Lin Xiao. Communication-efﬁcientdistributed optimization of self-concordant empiricalloss. arXiv preprint arXiv:1501.00263, 2015.[44] Yuchen Zhang, Martin J Wainwright, and John CDuchi. Communication-efﬁcient algorithms for statis-tical optimization. In NIPS, 2012.[45] Yuchen Zhang, John Duchi, Michael I Jordan, and Mar-tin J Wainwright. Information-theoretic lower boundsfor distributed statistical estimation with communica-tion constraints. In Advances in Neural InformationProcessing Systems, 2013.[46] Martin Zinkevich, Markus Weimer, Lihong Li, andAlex J. Smola. Parallelized stochastic gradient descent.In NIPS. 2010.
* [28] Ryan McDonald, Keith Hall, and Gideon Mann. Dis-tributed training strategies for the structured percep-tron. In NAACL HLT, 2010.[29] Natalia Neverova, Christian Wolf, Grifﬁn Lacey, LexFridman, Deepak Chandra, Brandon Barbello, andGraham W. Taylor. Learning human identity frommotion patterns. IEEE Access, 4:1810–1820, 2016.[30] Jacob Poushter. Smartphone ownership and internetusage continues to climb in emerging economies. PewResearch Center Report, 2016.[31] Daniel Povey, Xiaohui Zhang, and Sanjeev Khudanpur.Parallel training of deep neural networks with naturalgradient and parameter averaging. In ICLR WorkshopTrack, 2015.[32] William Shakespeare.The Complete Works ofWilliam Shakespeare. Publically available at https:
* //www.gutenberg.org/ebooks/100.
* [33] Ohad Shamir and Nathan Srebro. Distributed stochas-tic optimization and learning. In Communication, Con-trol, and Computing (Allerton), 2014.[34] Ohad Shamir, Nathan Srebro, and Tong Zhang. Com-munication efﬁcient distributed optimization usingan approximate newton-type method. arXiv preprintarXiv:1312.7853, 2013.[35] Reza Shokri and Vitaly Shmatikov. Privacy-preservingdeep learning.In Proceedings of the 22Nd ACMSIGSAC Conference on Computer and Communica-tions Security, CCS ’15, 2015.[36] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky,Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: Asimple way to prevent neural networks from overﬁtting.15, 2014.[37] Latanya Sweeney. Simple demographics often identifypeople uniquely. 2000.[38] TensorFlow team. Tensorﬂow convolutional neural net-
* works tutorial, 2016. http://www.tensorflow.org/tutorials/deep\_cnn.
* [39] White House Report. Consumer data privacy in a net-worked world: A framework for protecting privacy andpromoting innovation in the global digital economy.Journal of Privacy and Conﬁdentiality, 2013.[40] Tianbao Yang. Trading computation for communica-tion: Distributed stochastic dual coordinate ascent. InAdvances in Neural Information Processing Systems,2013.[41] Ruiliang Zhang and James Kwok. Asynchronous dis-tributed admm for consensus optimization. In ICML.JMLR Workshop and Conference Proceedings, 2014.[42] Sixin Zhang, Anna E Choromanska, and Yann LeCun.Deep learning with elastic averaging sgd. In NIPS.2015.
* H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, Blaise Ag ¨uera y Arcas
* A Supplemental Figures and Tables
* Table 4: Speedups in the number of communication roundsto reach a target accuracy of 97% for FedAvg, versusFedSGD (ﬁrst row) on the MNIST 2NN model.
* MN IST 2NN
* F E D SGD
* F E DA V GF E D A V GF E DA V GF E D A V GF E D A V GF E D A V GF E D A V GF E D A V G
* Figure 6: Training set convergence for the MNIST CNN.Note the y -axis is on a log scale, and the x-axis covers moretraining than Figure 2. These plots ﬁx C = 0.1.
* E
* B
* 1 ∞
* 10 ∞15020 ∞1101050205010102010
* u
* 1101220601202406001200
* I ID
* 1468
* 156 (9.4×)144 (10.2×)92 (16.0×)92 (16.0×)45 (32.6×)39 (37.6×)34 (43.2×)32 (45.9×)
* NON - I ID18171100 (1.7×)1183 (1.5×)
* 957 (1.9×)831 (2.2×)881 (2.1×)835 (2.2×)497 (3.7×)738 (2.5×)
* Figure 9: Test accuracy versus number of minibatch gradientcomputations (B = 50). The baseline is standard sequentialSGD, as compared to FedAvg with different client fractionsC (recall C = 0 means one client per round), and differentnumbers of local epochs E .
* Figure 7: Test set accuracy vs. communication rounds forMNIST 2NN with C = 0.1 and optimized η . The leftcolumn is the IID dataset, and right is the pathological 2-digits-per-client non-IID data.
* Figure 8: The effect of training for many local epochs (largeE ) between averaging steps, ﬁxing B = 10 and C = 0.1.Training loss for the MNIST CNN. Note different learningrates and y -axis scales are used due to the difﬁculty of ourpathological non-IID MNIST dataset.
* Figure 10: Learning curves for the large-scale languagemodel word LSTM, with evaluation computed every 20rounds. FedAvg actually performs better with fewer localepochs E (1 vs 5), and also has lower variance in accuracyacross evaluation rounds compared to FedSGD.