Communication-Efficient Learning of Deep Networks from Decentralized Data

H. Brendan McMahan **Eider Moore** **Daniel Ramage** **Seth Hampson** **Blaise Agüera y Arcas**
Google, Inc., 651 N 34th St., Seattle, WA 98103 USA

Abstract

Modern mobile devices have access to a wealth of data suitable for learning models, which in turn can greatly improve the user experience on the device. For example, language models can improve speech recognition and text entry, and image models can automatically select good photos. However, this rich data is often privacy sensitive, large in quantity, or both, which may preclude logging to the data center and training there using conventional approaches. We advocate an alternative that leaves the training data distributed on the mobile devices, and learns a shared model by aggregating locally-computed updates. We term this decentralized approach *Federated Learning*.

We present a practical method for the federated learning of deep networks based on iterative model averaging, and conduct an extensive empirical evaluation, considering five different model architectures and four datasets. These experiments demonstrate the approach is robust to the unbalanced and non-IID data distributions that are a defining characteristic of this setting. Communication costs are the principal constraint, and we show a reduction in required communication rounds by 10–100× as compared to synchronized stochastic gradient descent.

1 Introduction

Increasingly, phones and tablets are the primary computing devices for many people [30, 2]. The powerful sensors on these devices (including cameras, microphones, and GPS), combined with the fact they are frequently carried, means they have access to an unprecedented amount of data, much of it private in nature. Models learned on such data hold the promise of greatly improving usability by powering more intelligent applications, but the sensitive nature of the data means there are risks and responsibilities to storing it in a centralized location.

We investigate a learning technique that allows users to collectively reap the benefits of shared models trained from this rich data, without the need to centrally store it. We term our approach *Federated Learning*, since the learning task is solved by a loose federation of participating devices (which we refer to as *clients*) which are coordinated by a central *server*. Each client has a local training dataset which is never uploaded to the server. Instead, each client computes an update to the current global model maintained by the server, and only this update is communicated. This is a direct application of the principle of *focused collection* or *data minimization* proposed by the 2012 White House report on privacy of consumer data [39]. Since these updates are specific to improving the current model, there is no reason to store them once they have been applied.

A principal advantage of this approach is the decoupling of model training from the need for direct access to the raw training data. Clearly, some trust of the server coordinating the training is still required. However, for applications where the training objective can be specified on the basis of data available on each client, federated learning can significantly reduce privacy and security risks by limiting the attack surface to only the device, rather than the device and the cloud.

Our primary contributions are 1) the identification of the problem of training on decentralized data from mobile devices as an important research direction; 2) the selection of a straightforward and practical algorithm that can be applied to this setting; and 3) an extensive empirical evaluation of the proposed approach. More concretely, we introduce the `FederatedAveraging` algorithm, which combines local stochastic gradient descent (SGD) on each client with a server that performs model averaging. We perform extensive experiments on this algorithm, demonstrating it is robust to unbalanced and non-IID data distributions, and can reduce the rounds of communication needed to train a deep network on decentralized data by orders of magnitude.

Appearing in Proceedings of the 20^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2017, Fort Lauderdale, Flordia, USA. JMLR: W&CP volume 54.

深度网络的通信高效学习分散式数据

H.布伦丹·麦克马汉·艾德摩尔丹尼尔·拉梅奇·塞思·汉普森·布莱斯·阿古埃拉·阿卡斯
谷歌公司，651 N 34th St.，美国华盛顿州西雅图，邮编98103

摘要

现代移动的设备可以访问大量适合学习模型的数据，这反过来又可以大大改善设备上的用户体验。例如，语言模型可以改善语音识别和文本输入，图像模型可以自动选择好的照片。然而，这种丰富的数据通常是隐私敏感的、数量大的或两者兼而有之，这可能妨碍使用常规方法记录到数据中心并在那里进行训练。我们提倡一种替代方案，将训练数据分布在移动的设备上，并通过聚合本地计算的更新来学习共享模型。我们将这种分散的方法称为联邦学习。

我们提出了一种基于迭代模型平均的深度网络联合学习的实用方法，并进行了广泛的实证评估，考虑了五种不同的模型架构和四个数据集。这些实验表明，该方法对不平衡和非IID数据分布是鲁棒的，这是该设置的定义特征。通信成本是主要的约束条件，与同步随机梯度下降相比，我们显示出所需的通信轮次减少了10 - 100 倍。

1 介绍

手机和平板电脑越来越多地成为许多人的主要计算设备[30，2]。这些设备上强大的传感器（包括摄像头、麦克风和GPS），加上它们经常携带的事实，意味着它们可以访问前所未有的大量数据，其中大部分是私人数据。在这些数据上学习的模型通过为更智能的应用程序提供支持，有望大大提高可用性，但数据的敏感性意味着将其存储在集中位置存在风险和责任。

我们研究了一种学习技术，它允许用户集体获得从这些丰富的数据中训练的共享模型的好处，而不需要集中存储它。我们称我们的方法为联合学习，因为学习任务是由一个松散的联合参与设备（我们称之为客户端），由一个中央服务器协调解决的。每个客户端都有一个本地训练数据集，它永远不会上传到服务器。相反，每个客户端计算对服务器维护的当前全局模型的更新，并且仅传递此更新。这是对2012 年白宫关于消费者数据隐私的报告提出的集中收集或数据最小化原则的直接应用[39]。由于这些更新是特定于改进当前模型的，因此一旦应用它们，就没有理由存储它们。

这种方法的主要优点是将模型训练与直接访问原始训练数据的需求解耦。显然，仍然需要对协调训练的服务器有一定的信任。然而，对于可以根据每个客户端上的可用数据指定训练目标的应用程序，联邦学习可以通过将攻击面限制在设备上而不是设备和云来显著降低隐私和安全风险。

我们的主要贡献是：1）识别来自移动的设备的分散数据的训练问题作为一个重要的研究方向; 2）选择一个简单实用的算法，可以应用于此设置;和3）所提出的方法的广泛的实证评估。更具体地说，我们引入了FederatedAveraging 算法，该算法将每个客户端上的局部随机梯度下降（SGD）与执行模型平均的服务器相结合。我们对该算法进行了广泛的实验，证明它对不平衡和非IID数据分布具有鲁棒性，并且可以将在分散数据上训练深度网络所需的通信次数减少几个数量级。

出现在2017 年人工智能和统计国际会议（AISTATS）的会议记录中，劳德代尔堡，佛罗里达州，美国。JMLR：W&CP卷54 。

Federated Learning Ideal problems for federated learning have the following properties: 1) Training on real-world data from mobile devices provides a distinct advantage over training on proxy data that is generally available in the data center. 2) This data is privacy sensitive or large in size (compared to the size of the model), so it is preferable not to log it to the data center purely for the purpose of model training (in service of the *focused collection* principle). 3) For supervised tasks, labels on the data can be inferred naturally from user interaction.

Many models that power intelligent behavior on mobile devices fit the above criteria. As two examples, we consider *image classification*, for example predicting which photos are most likely to be viewed multiple times in the future, or shared; and *language models*, which can be used to improve voice recognition and text entry on touch-screen keyboards by improving decoding, next-word-prediction, and even predicting whole replies [10]. The potential training data for both these tasks (all the photos a user takes and everything they type on their mobile keyboard, including passwords, URLs, messages, etc.) can be privacy sensitive. The distributions from which these examples are drawn are also likely to differ substantially from easily available proxy datasets: the use of language in chat and text messages is generally much different than standard language corpora, e.g., Wikipedia and other web documents; the photos people take on their phone are likely quite different than typical Flickr photos. And finally, the labels for these problems are directly available: entered text is self-labeled for learning a language model, and photo labels can be defined by natural user interaction with their photo app (which photos are deleted, shared, or viewed).

Both of these tasks are well-suited to learning a neural network. For image classification feed-forward deep networks, and in particular convolutional networks, are well-known to provide state-of-the-art results [26, 25]. For language modeling tasks recurrent neural networks, and in particular LSTMs, have achieved state-of-the-art results [20, 5, 22].

Privacy Federated learning has distinct privacy advantages compared to data center training on persisted data. Holding even an "anonymized" dataset can still put user privacy at risk via joins with other data [37]. In contrast, the information transmitted for federated learning is the minimal update necessary to improve a particular model (naturally, the strength of the privacy benefit depends on the content of the updates).[1] The updates themselves can (and should) be ephemeral. They will never contain more information than the raw training data (by the data processing inequality), and will generally contain much less. Further, the source of the updates is not needed by the aggregation algorithm, so updates can be transmitted without identifying meta-data over a mix network such as Tor [7] or via a trusted third party. We briefly discuss the possibility of combining federated learning with secure multiparty computation and differential privacy at the end of the paper.

[1] For example, if the update is the total gradient of the loss on all of the local data, and the features are a sparse bag-of-words, then the non-zero gradients reveal exactly which words the user has entered on the device. In contrast, the sum of many gradients for a dense model such as a CNN offers a harder target for attackers seeking information about individual training instances (though attacks are still possible).

Federated Optimization We refer to the optimization problem implicit in federated learning as federated optimization, drawing a connection (and contrast) to distributed optimization. Federated optimization has several key properties that differentiate it from a typical distributed optimization problem:

- **Non-IID** The training data on a given client is typically based on the usage of the mobile device by a particular user, and hence any particular user's local dataset will not be representative of the population distribution.
- **Unbalanced** Similarly, some users will make much heavier use of the service or app than others, leading to varying amounts of local training data.
- **Massively distributed** We expect the number of clients participating in an optimization to be much larger than the average number of examples per client.
- **Limited communication** Mobile devices are frequently offline or on slow or expensive connections.

In this work, our emphasis is on the non-IID and unbalanced properties of the optimization, as well as the critical nature of the communication constraints. A deployed federated optimization system must also address a myriad of practical issues: client datasets that change as data is added and deleted; client availability that correlates with the local data distribution in complex ways (e.g., phones from speakers of American English will likely be plugged in at different times than speakers of British English); and clients that never respond or send corrupted updates.

These issues are beyond the scope of the current work; instead, we use a controlled environment that is suitable for experiments, but still addresses the key issues of client availability and unbalanced and non-IID data. We assume a synchronous update scheme that proceeds in rounds of communication. There is a fixed set of K clients, each with a fixed local dataset. At the beginning of each round, a random fraction C of clients is selected, and the server sends the current global algorithm state to each of these clients (e.g., the current model parameters). We only select a fraction of clients for efficiency, as our experiments show diminishing returns for adding more clients beyond a certain point. Each selected client then performs local computation based on the global state and its local dataset, and sends an update to the server. The server then applies these updates to its global state, and the process repeats.

联邦学习联邦学习的理想问题具有以下属性：1）在来自移动的设备的真实数据上进行训练，与在数据中心通常可用的代理数据上进行训练相比，具有明显的优势。2)这些数据是隐私敏感的，或者数据量很大（与模型的大小相比），因此最好不要将其记录到数据中心，纯粹用于模型训练（服务于集中收集原则）。3)对于监督任务，数据上的标签可以从用户交互中自然推断出来。

许多在移动的设备上支持智能行为的模型都符合上述标准。作为两个例子，我们考虑图像分类，例如预测哪些照片最有可能在未来被多次查看或共享;语言模型，可用于通过改进解码，下一个单词预测甚至预测整个回复来改进语音识别和触摸屏键盘上的文本输入。这两项任务的潜在训练数据（用户拍摄的所有照片以及他们在移动的键盘上键入的所有内容，包括密码、URL、消息等）可以是隐私敏感的。这些例子的分布也可能与容易获得的代理数据集有很大的不同：聊天和短信中的语言使用通常与标准语言语料库有很大的不同，例如。维基百科和其他网络文档:人们在手机上拍摄的照片可能与典型的Flickr照片大不相同。最后，这些问题的标签是直接可用的：输入的文本是自标记的，用于学习语言模型，照片标签可以通过用户与照片应用程序的自然交互来定义（删除、共享或查看哪些照片）。

这两个任务都非常适合学习神经网络。对于图像分类，前馈深度网络，特别是卷积网络，众所周知可以提供最先进的结果[26，25]。对于语言建模任务，递归神经网络，特别是LSTM，已经取得了最先进的结果[20，5，22]。

与数据中心对持久化数据的训练相比，联合学习具有明显的隐私优势。即使持有“ 匿名” 数据集，也可能通过与其他数据的连接使用户隐私处于危险之中[37]。相比之下，为联邦学习传输的信息是改进特定模型所需的最小更新（自然，隐私优势的强度取决于更新的内容）。更新本身可以（也应该）是短暂的。他们永远不会包含更多的信息-

例如，如果更新是所有本地数据的丢失的总梯度，并且特征是稀疏的词袋，则非零梯度准确地揭示了用户在设备上输入了哪些词。相比之下，密集模型（如CNN ）的许多梯度之和为寻求有关单个训练实例的信息的攻击者提供了更难的目标（尽管攻击仍然是可能的）。

此外，聚合算法不需要更新的来源，因此更新可以在不识别元数据的情况下通过混合网络（如Tor [7]）或通过可信第三方传输。在文章的最后，我们简要讨论了将联邦学习与安全多方计算和差分隐私相结合的可能性。

联邦优化我们将联邦学习中隐含的优化问题称为联邦优化，与分布式优化有联系（和对比）。联邦优化有几个关键属性，将其与典型的分布式优化问题区分开来：

- 非IID给定客户端上的训练数据通常基于特定用户对移动终端的使用，因此任何特定用户的本地数据集将不代表群体分布。
- 同样，一些用户会比其他用户更频繁地使用服务或应用程序，从而导致本地训练数据量的变化。
- 大规模分布式我们期望参与优化的客户端数量远大于每个客户端的平均示例数量。
- 有限的通信移动的设备经常离线或处于缓慢或昂贵的连接上。

在这项工作中，我们的重点是非IID和不平衡的优化性能，以及通信约束的关键性质。部署的联邦优化系统还必须解决无数的实际问题：随着数据的添加和删除而改变的客户端数据集;以复杂方式与本地数据分布相关的客户端可用性（例如，来自美国英语使用者的电话可能会在与英国英语使用者不同的时间插入）;以及从不响应或发送损坏的更新的客户端。

这些问题超出了当前工作的范围:相反，我们使用适合实验的受控环境，但仍然解决了客户端可用性和不平衡和非IID数据的关键问题。我们假设一个同步更新方案，在通信轮进行。有一组固定的K个客户端，每个客户端都有一个固定的本地数据集。在每轮开始时，选择客户端的随机部分C，并且服务器将当前全局算法状态发送到这些客户端中的每一个（例如，当前模型参数）。我们只选择一小部分客户端来提高效率，因为我们的实验表明，超过一定程度，增加更多客户端的回报会减少。然后，每个选定的客户端基于全局状态及其本地数据集执行本地计算，并向服务器发送更新。然后，服务器将这些更新应用于其全局状态，并重复该过程。

While we focus on non-convex neural network objectives, the algorithm we consider is applicable to any finite-sum objective of the form

$$\min_{w \in \mathbb{R}^d} f(w) \qquad \text{where} \qquad f(w) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} f_i(w). \quad (1)$$

For a machine learning problem, we typically take $f_i(w) = \ell(x_i, y_i; w)$, that is, the loss of the prediction on example (x_i, y_i) made with model parameters w. We assume there are K clients over which the data is partitioned, with $\mathcal{P}_k$ the set of indexes of data points on client k, with $n_k = |\mathcal{P}_k|$. Thus, we can re-write the objective (1) as

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \quad \text{where} \quad F_k(w) = \frac{1}{n_k} \sum_{i \in \mathcal{P}_k} f_i(w).$$

If the partition $\mathcal{P}_k$ was formed by distributing the training examples over the clients uniformly at random, then we would have $\mathbb{E}_{\mathcal{P}_k}[F_k(w)] = f(w)$, where the expectation is over the set of examples assigned to a fixed client k. This is the IID assumption typically made by distributed optimization algorithms; we refer to the case where this does not hold (that is, F_k could be an arbitrarily bad approximation to f) as the non-IID setting.

In data center optimization, communication costs are relatively small, and computational costs dominate, with much of the recent emphasis being on using GPUs to lower these costs. In contrast, in federated optimization communication costs dominate — we will typically be limited by an upload bandwidth of 1 MB/s or less. Further, clients will typically only volunteer to participate in the optimization when they are charged, plugged-in, and on an unmetered wi-fi connection. Further, we expect each client will only participate in a small number of update rounds per day. On the other hand, since any single on-device dataset is small compared to the total dataset size, and modern smartphones have relatively fast processors (including GPUs), computation becomes essentially free compared to communication costs for many model types. Thus, our goal is to use additional computation in order to decrease the number of rounds of communication needed to train a model. There are two primary ways we can add computation: 1) *increased parallelism*, where we use more clients working independently between each communication round; and, 2) *increased computation on each client*, where rather than performing a simple computation like a gradient calculation, each client performs a more complex calculation between each communication round. We investigate both of these approaches, but the speedups we achieve are due primarily to adding more computation on each client, once a minimum level of parallelism over clients is used.

Related Work Distributed training by iteratively averaging locally trained models has been studied by McDonald et al. [28] for the perceptron and Povey et al. [31] for speech recognition DNNs. Zhang et al. [42] studies an asynchronous approach with "soft" averaging. These works only consider the cluster / data center setting (at most 16 workers, wall-clock time based on fast networks), and do not consider datasets that are unbalanced and non-IID, properties that are essential to the federated learning setting. We adapt this style of algorithm to the federated setting and perform the appropriate empirical evaluation, which asks different questions than those relevant in the data center setting, and requires different methodology.

Using similar motivation to ours, Neverova et al. [29] also discusses the advantages of keeping sensitive user data on device. The work of Shokri and Shmatikov [35] is related in several ways: they focus on training deep networks, emphasize the importance of privacy, and address communication costs by only sharing a subset of the parameters during each round of communication; however, they also do not consider unbalanced and non-IID data, and the empirical evaluation is limited.

In the convex setting, the problem of distributed optimization and estimation has received significant attention [4, 15, 33], and some algorithms do focus specifically on communication efficiency [45, 34, 40, 27, 43]. In addition to assuming convexity, this existing work generally requires that the number of clients is much smaller than the number of examples per client, that the data is distributed across the clients in IID fashion, and that each node has an identical number of data points — all of these assumptions are violated in the federated optimization setting. Asynchronous distributed forms of SGD have also been applied to training neural networks, e.g., Dean et al. [12], but these approaches require a prohibitive number of updates in the federated setting. Distributed consensus algorithms (e.g., [41]) relax the IID assumption, but are still not a good fit for communication-constrained optimization over very many clients.

One endpoint of the (parameterized) algorithm family we consider is simple one-shot averaging, where each client solves for the model that minimizes (possibly regularized) loss on their local data, and these models are averaged to produce the final global model. This approach has been studied extensively in the convex case with IID data, and it is known that in the worst-case, the global model produced is no better than training a model on a single client [44, 3, 46].

2 The `FederatedAveraging` Algorithm

The recent multitude of successful applications of deep learning have almost exclusively relied on variants of stochastic gradient descent (SGD) for optimization; in fact, many advances can be understood as adapting the structure of the model (and hence the loss function) to be more amenable to optimization by simple gradient-based methods [16]. Thus, it is natural that we build algorithms for

虽然我们专注于非凸神经网络目标，但我们考虑的算法适用于以下形式的任何有限和目标

$$\min_{w\in\mathbb{R}} f(w) \text{ 其中} f(w) = \frac{1}{n}\sum_{i=1}^{n} f(w) 。 \quad (一)$$

对于一个机器学习问题，我们通常取f（w）= `（x，y; w），即用模型参数w对示例（x，y）进行预测的损失。我们假设有K个客户端，数据在这些客户端上被分区，其中P是客户端k上的数据点的索引集合，其中n =| P|.因此，我们可以将目标（1）重写为

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) ，\text{其中} F_k(w) = \frac{1}{n_k}\sum_{i\in\mathcal{P}_k} f(w) 。$$

如果划分P是通过将训练样本随机均匀地分布在客户端上而形成的，那么我们将得到E[F（w）] = f（w），其中期望是分配给固定客户端k的样本集。这是分布式优化算法通常所做的IID假设;我们将不成立的情况（即F可能是f的任意差近似）称为非IID设置。

在数据中心优化中，通信成本相对较小，计算成本占主导地位，最近的重点是使用GPU来降低这些成本。相比之下，在联合优化中，通信成本占主导地位--我们通常会受到1 MB/s或更少的上传带宽的限制。此外，客户端通常只会在收费、插入和使用未计量的wi-fi连接时自愿参与优化。此外，我们预计每个客户端每天只会参与少量的更新轮。另一方面，由于任何单个设备上的数据集与总数据集大小相比都很小，并且现代智能手机具有相对快速的处理器（包括GPU），因此与许多模型类型的通信成本相比，计算基本上是免费的。因此，我们的目标是使用额外的计算，以减少训练模型所需的通信轮数。 我们可以通过两种主要方式来增加计算：1）增加并行性，即在每个通信回合之间使用更多独立工作的客户端; 2）增加每个客户端上的计算，即每个客户端在每个通信回合之间执行更复杂的计算，而不是执行简单的计算（如梯度计算）。我们调查这两种方法，但我们实现的加速主要是由于增加了更多的计算在每个客户端上，一旦最低水平的并行客户端使用。

McDonald 等人[28]针对感知器和Povey 等人[29]研究了通过迭代平均本地训练模型进行分布式训练的方法。

[31]语音识别DNN Zhang 等人[42]研究了一种具有“ 软”平均的异步方法。这些工作只考虑集群/数据中心设置（最多16 个工作人员，基于快速网络的挂钟时间），而不考虑不平衡和非IID的数据集，这些属性对联合学习设置至关重要。我们将这种风格的算法适应于联邦设置，并进行适当的经验评估，这会提出与数据中心设置中相关的问题不同的问题，并且需要不同的方法。

使用与我们类似的动机，Neverova 等人。[29]还讨论了将敏感用户数据保存在设备上的优势。Shokri 和Shmatikov [35]的工作在几个方面相关：他们专注于训练深度网络，强调隐私的重要性，并通过在每一轮通信期间仅共享参数的子集来解决通信成本;然而，他们也没有考虑不平衡和非IID数据，并且经验评估有限。

在凸设置中，分布式优化和估计的问题受到了极大的关注[4，15，33]，并且一些算法确实特别关注通信效率[45，34，40，27，43]。除了假设凸性之外，现有的工作通常要求客户端的数量远小于每个客户端的示例数量，数据以IID方式分布在客户端之间，并且每个节点具有相同数量的数据点-所有这些假设都违反了联合优化设置。SGD的异步分布式形式也已被应用于训练神经网络，例如，Dean 等人[12]，但是这些方法需要在联邦设置中进行大量的更新。分布式一致性算法（例如，[41]）放松了IID假设，但仍然不适合在非常多的客户端上进行通信约束优化。

我们考虑的（参数化）算法家族的一个端点是简单的单次平均，其中每个客户端求解最小化（可能正则化）其本地数据损失的模型，并对这些模型进行平均以产生最终的全局模型。这种方法已经在IID数据的凸情况下进行了广泛的研究，并且已知在最坏的情况下，生成的全局模型并不比在单个客户端上训练模型更好[44，3，46]。

2联合平均算法

最近大量成功的深度学习应用几乎完全依赖于随机梯度下降（SGD）的变体进行优化;事实上，许多进步可以理解为调整模型的结构（以及损失函数），使其更适合于通过简单的基于梯度的方法进行优化[16]。因此，我们很自然地建立算法，

federated optimization by starting from SGD.

SGD can be applied naively to the federated optimization problem, where a single batch gradient calculation (say on a randomly selected client) is done per round of communication. This approach is computationally efficient, but requires very large numbers of rounds of training to produce good models (e.g., even using an advanced approach like batch normalization, Ioffe and Szegedy [21] trained MNIST for 50000 steps on minibatches of size 60). We consider this baseline in our CIFAR-10 experiments.

In the federated setting, there is little cost in wall-clock time to involving more clients, and so for our baseline we use large-batch synchronous SGD; experiments by Chen et al. [8] show this approach is state-of-the-art in the data center setting, where it outperforms asynchronous approaches. To apply this approach in the federated setting, we select a C-fraction of clients on each round, and compute the gradient of the loss over all the data held by these clients. Thus, C controls the *global* batch size, with $C = 1$ corresponding to full-batch (non-stochastic) gradient descent.[2] We refer to this baseline algorithm as `FederatedSGD` (or `FedSGD`).

A typical implementation of `FedSGD` with $C = 1$ and a fixed learning rate η has each client k compute $g_k = \nabla F_k(w_t)$, the average gradient on its local data at the current model w_t, and the central server aggregates these gradients and applies the update $w_{t+1} \leftarrow w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k$, since $\sum_{k=1}^{K} \frac{n_k}{n} g_k = \nabla f(w_t)$. An equivalent update is given by $\forall k,\ w_{t+1}^k \leftarrow w_t - \eta g_k$ and then $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$. That is, each client locally takes one step of gradient descent on the current model using its local data, and the server then takes a weighted average of the resulting models. Once the algorithm is written this way, we can add more computation to each client by iterating the local update $w^k \leftarrow w^k - \eta \nabla F_k(w^k)$ multiple times before the averaging step. We term this approach `FederatedAveraging` (or `FedAvg`). The amount of computation is controlled by three key parameters: C, the fraction of clients that perform computation on each round; E, then number of training passes each client makes over its local dataset on each round; and B, the local minibatch size used for the client updates. We write $B = \infty$ to indicate that the full local dataset is treated as a single minibatch. Thus, at one endpoint of this algorithm family, we can take $B = \infty$ and $E = 1$ which corresponds exactly to `FedSGD`. For a client with n_k local examples, the number of local updates per round is given by $u_k = E\frac{n_k}{B}$; Complete pseudo-code is given in Algorithm 1.

For general non-convex objectives, averaging models in parameter space could produce an arbitrarily bad model.

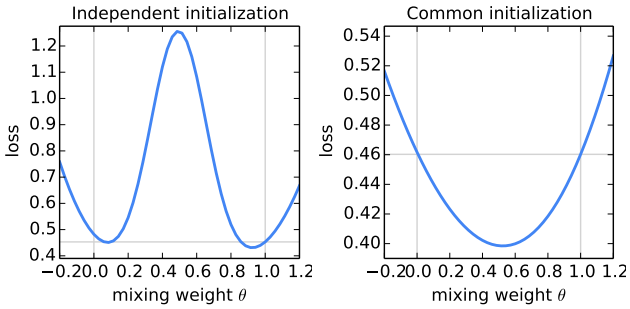


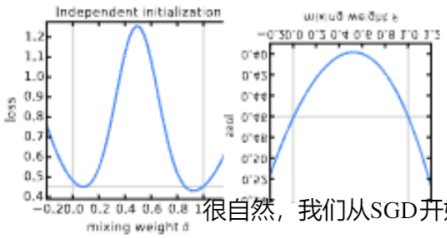
Figure 1: The loss on the full MNIST training set for models generated by averaging the parameters of two models w and w' using $\theta w + (1-\theta)w'$ for 50 evenly spaced values $\theta \in [-0.2, 1.2]$.The models w and w' were trained using SGD on different small datasets. For the left plot, w and w' were initialized using different random seeds; for the right plot, a shared seed was used. Note the different y-axis scales. The horizontal line gives the best loss achieved by w or w' (which were quite close, corresponding to the vertical lines at $\theta = 0$ and $\theta = 1$). With shared initialization, averaging the models produces a significant reduction in the loss on the total training set (much better than the loss of either parent model).

Following the approach of Goodfellow et al. [17], we see exactly this bad behavior when we average two MNIST digit-recognition models[3] trained from different initial conditions (Figure 1, left). For this figure, the parent models w and w' were each trained on non-overlapping IID samples of 600 examples from the MNIST training set. Training was via SGD with a fixed learning rate of 0.1 for 240 updates on minibatches of size 50 (or $E = 20$ passes over the mini-datasets of size 600). This is approximately the amount of training where the models begin to overfit their local datasets.

Recent work indicates that in practice, the loss surfaces of sufficiently over-parameterized NNs are surprisingly well-behaved and in particular less prone to bad local minima than previously thought [11, 17, 9]. And indeed, when we start two models *from the same random initialization* and then again train each independently on a different subset of the data (as described above), we find that naive parameter averaging works surprisingly well (Figure 1, right): the average of these two models, $\frac{1}{2}w + \frac{1}{2}w'$, achieves significantly lower loss on the full MNIST training set than the best model achieved by training on either of the small datasets independently. While Figure 1 starts from a random initialization, note a shared starting model w_t is used for each round of `FedAvg`, and so the same intuition applies.

[2] While the batch selection mechanism is different than selecting a batch by choosing individual examples uniformly at random, the batch gradients g computed by `FedSGD` still satisfy $\mathbb{E}[g] = \nabla f(w)$.

[3] We use the "2NN" multi-layer perceptron described in Section 3.



很自然，我们从SGD开始构建联邦优化算法。

SGD可以简单地应用于联邦优化问题，其中每一轮通信都要完成一次批量梯度计算（比如在随机选择的客户端上）。这种方法在计算上是高效的，但是需要非常大量的训练轮次来产生良好的模型（例如，即使使用像批量归一化这样的高级方法，Ioffe 和Szegedy [21]也在大小为60 的小批量上训练了50000 步的MNIST）。我们在CIFAR-10 实验中考虑了这一基线。

在联邦设置中，涉及更多客户端的挂钟时间成本很小，因此对于我们的基线，我们使用大批量同步SGD; Chen 等人的实验[8]表明这种方法在数据中心设置中是最先进的，它优于异步方法。为了在联邦设置中应用这种方法，我们在每一轮中选择Cfraction 客户端，并计算这些客户端持有的所有数据的损失梯度。因此，C控制全局批量大小，C = 1对应于全批量（非随机）梯度下降。我们将此基线算法称为FederatedSGD （或FedSGD ）。

具有C = 1和固定学习率η的FedSGD 的典型实现使每个客户端k计算g = OF（w），即当前模型w处其本地数据的平均梯度，并且中央服务器聚合这些梯度并应用更新w <$w − η $\sum_{k=1}^{n} \frac{n}{n} g$，因为 $\sum_{k=1}^{n} \frac{n}{n} Of\ ^{n}(w)$。一个等价的更新由下式给出：$\sum_{k=1}^{n} \frac{n}{n} w.$

也就是说，每个客户端使用其本地数据对当前模型进行一步梯度下降，然后服务器对结果模型进行加权平均。一旦算法以这种方式编写，我们就可以通过在平均步骤之前多次迭代本地更新w <$w − ηOF（w）来为每个客户端增加更多的计算。我们称这种方法为FederatedAveraging （或FedAvg ）。计算量由三个关键参数控制：C，每轮执行计算的客户端的比例; E，每个客户端在每轮对其本地数据集进行训练的次数;以及B，用于客户端更新的本地小批量大小。我们写B = ∞来表示整个局部数据集被视为单个小批。因此，在这个算法族的一个端点，我们可以取B = ∞和E = 1，这正好对应于FedSGD 。 对于具有nlocal 示例的客户端，每轮本地更新的数量由u = E给出;算法1 中给出了完整的伪代码。

对于一般的非凸目标，在参数空间中平均模型可以产生任意坏的模型。

图一：模型的完整MNIST 训练集上的损失，通过对两个模型w和w的参数进行平均来生成，使用θw + （1 − θ）w for 50 个均匀间隔的值θ ∈ [−0.2，1.2]。模型w和w使用SGD在不同的小数据集上进行训练。对于左图，w和w使用不同的随机种子初始化;对于右图，使用共享种子。注意不同的y轴刻度。水平线给出了w或w（它们非常接近，对应于θ = 0和θ = 1时的垂直线）实现的最佳损失。通过共享初始化，平均模型可以显著减少总训练集的损失（比任何一个父模型的损失都要好）。

遵循Goodfellow 等人的方法。[17]，当我们对两个从不同初始条件应变的MNIST 数字识别模型进行平均时，我们正好看到了这种不良行为（图1，左）。对于该图，父模型w和w分别在来自MNIST 训练集的600 个示例的非重叠IID样本上进行训练。训练是通过SGD进行的，固定学习率为0.1，在大小为50 的小批量上进行240 次更新（或者在大小为600 的小数据集上进行E = 20 次）。这大约是模型开始过拟合其局部数据集的训练量。

最近的工作表明，在实践中，充分过参数化的NN 的损失面表现得令人惊讶，特别是不像以前认为的那样容易出现坏的局部极小值[11，17，9]。事实上，当我们从相同的随机初始化开始两个模型，然后在不同的数据子集上分别训练每个模型时，（如上所述），我们发现朴素的参数平均效果令人惊讶地好（图1，右）：这两个模型的平均值w + w在完整的MNIST 训练集上实现的损失显著低于通过独立地在任何一个小数据集上训练实现的最佳模型。虽然图1是从随机初始化开始的，但请注意，每一轮FedAvg 都使用了一个共享的起始模型，因此同样的直觉也适用。

虽然批次选择机制不同于通过随机均匀地选择单个示例来选择批次，但FedSGD 计算的批次梯度g仍然满足E[g] = Of （w）。

3我们使用第3节中描述的“ 2NN ” 多层感知器。

Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:
initialize w_0
for each round $t = 1, 2, \dots$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of m clients)
  for each client $k \in S_t$ **in parallel do**
    $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
  $m_t \leftarrow \sum_{k \in S_t} n_k$
  $w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$ *// Erratum*[4]

ClientUpdate(k, w): *// Run on client k*
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
for each local epoch i from 1 to E **do**
  for batch $b \in \mathcal{B}$ **do**
    $w \leftarrow w - \eta \nabla \ell(w; b)$
return w to server

3 Experimental Results

We are motivated by both image classification and language modeling tasks where good models can greatly enhance the usability of mobile devices. For each of these tasks we first picked a proxy dataset of modest enough size that we could thoroughly investigate the hyperparameters of the `FedAvg` algorithm. While each individual training run is relatively small, we trained over 2000 individual models for these experiments. We then present results on the benchmark CIFAR-10 image classification task. Finally, to demonstrate the effectiveness of `FedAvg` on a real-world problem with a natural partitioning of the data over clients, we evaluate on a large language modeling task.

Our initial study includes three model families on two datasets. The first two are for the MNIST digit recognition task [26]: 1) A simple multilayer-perceptron with 2-hidden layers with 200 units each using ReLu activations (199,210 total parameters), which we refer to as the MNIST 2NN. 2) A CNN with two 5x5 convolution layers (the first with 32 channels, the second with 64, each followed with 2x2 max pooling), a fully connected layer with 512 units and ReLu activation, and a final softmax output layer (1,663,370 total parameters). To study federated optimization, we also need to specify how the data is distributed over the clients. We study two ways of partitioning the MNIST data over clients: **IID**, where the data is shuffled, and then partitioned into 100 clients each receiving 600 examples, and **Non-IID**, where we first *sort* the data by digit label, divide it into 200 shards of size 300, and assign each of 100 clients 2 shards. This is a pathological non-IID partition of the data, as most clients will only have examples of two digits, letting us explore the degree to which our algorithms will break on highly non-IID data. Both of these partitions are balanced, however.[5]

For language modeling, we built a dataset from *The Complete Works of William Shakespeare* [32]. We construct a client dataset for each speaking role in each play with at least 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 last 20%, rounded up to at least one line). The resulting dataset has 3,564,579 characters in the training set, and 870,014 characters[6] in the test set. This data is substantially unbalanced, with many roles having only a few lines, and a few with a large number of lines. Further, observe the test set is not a random sample of lines, but is temporally separated by the chronology of each play. Using an identical train/test split, 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 language model, which after reading each character in a line, predicts the next character [22]. The model takes a series of characters as input and embeds each of these into a learned 8 dimensional space. The embedded characters are then processed through 2 LSTM layers, each with 256 nodes. Finally the output of the second LSTM layer is sent to a softmax output layer with one node per character. The full model has 866,578 parameters, and we trained using an unroll length of 80 characters.

SGD is sensitive to the tuning of the learning-rate parameter η. The results reported here are based on training over a sufficiently wide grid of learning rates (typically 11-13 values for η on a multiplicative grid of resolution $10^{\frac{1}{3}}$ or $10^{\frac{1}{6}}$). We checked to ensure the best learning rates were in the middle of our grids, and that there was not a significant difference between the best learning rates. Unless otherwise noted, we plot metrics for the best performing rate selected individually for each x-axis value. We find that the optimal learning rates do not vary too much as a function of the other parameters.

Increasing parallelism We first experiment with the client fraction C, which controls the amount of multi-client parallelism. Table 1 shows the impact of varying C for both MNIST models. We report the number of communication rounds necessary to achieve a target test-set accuracy. To compute this, we construct a learning curve for each combination of parameter settings, optimizing η as described above and then making each curve monotonically improving by taking the best value of test-set accuracy achieved over

[4] Earlier versions of this paper incorrectly indicated summation over all K clients here.

[5] We performed additional experiments on unbalanced versions of these datasets, and found them to in fact be slightly easier for `FedAvg`.

[6] We always use character to refer to a one byte string, and use role to refer to a part in the play.

算法1联合平均。K个客户端由k索引; B是本地minibatch 大小，E是本地epoch 的数量，η是学习率。

服务器执行:
对于每一轮t = 1，2，...做
m ← max（C · K，1）
S←（随机的m 个客户端集合）
对于每个客户端**k** ∈ **Sin**并行做
客户端更新Client Update （k，w）
$m \leftarrow \sum_{k \in Sn}$
$w \leftarrow \sum_{k \in S}$ ℏnw //*erratum* 错误

ClientUpdate （**k**，**w**）：//在客户端上运行**k**
B ←（将平托分成大小为B的批次）对
于从1到E的每个局部时期i，
对于批次**b** ∈ **B**，
w ← w − ηO`（w; ℬ)
返回w到服务器

3实验结果

我们的动机是图像分类和语言建模任务，其中好的模型可以大大提高移动的设备的可用性。对于这些任务中的每一个，我们首先选择一个足够大小的代理数据集，以便我们可以彻底研究FedAvg 算法的超参数。虽然每个单独的训练运行相对较小，但我们为这些实验训练了2000 多个单独的模型。然后，我们提出的基准CIFAR-10 图像分类任务的结果。最后，为了证明FedAvg 在实际问题上的有效性，我们对一个大型语言建模任务进行了评估，其中数据在客户端上进行了自然分区。

我们的初步研究包括两个数据集上的三个模型家族。前两个用于MNIST 数字识别任务[26]：1）一个简单的多层感知器，具有2 个隐藏层，每个隐藏层有200 个单元，使用ReLu 激活（199，210 个总参数），我们称之为MNIST 2NN 。2)CNN 有两个5x 5卷积层（第一个有32 个通道，第二个有64 个通道，每个都有2x2 最大池），一个有512 个单元和ReLu激活的全连接层，以及一个最终的softmax 输出层（总共1，663，370 个参数）。为了研究联邦优化，我们还需要指定数据如何在客户机上分布。我们研究了在客户端上划分MNIST 数据的两种方法：IID，其中数据被打乱，然后划分为100 个客户端，每个客户端接收600 个示例，以及Non - IID，其中我们首先按数字标签对数据进行排序，将其划分为200 个大小为300 的分片，并为100 个客户端中的每个客户端分配2 个分片。 这是一种病态的非IID数据分区，因为大多数客户端

4本文的早期版本错误地指出了这里所有K个客户端的求和。

将只有两位数的例子，让我们探索我们的算法在高度非IID数据上的破坏程度。然而，这两个分区是平衡的。

对于语言建模，我们从The Com 构建了一个数据集，

威廉·莎士比亚全集[32]我们为每个角色构建一个客户端数据集，每个角色至少有两行。这产生了具有1146 个客户端的数据集。对于每个客户端，我们将数据分为一组训练行（前80 %的行用于角色）和测试行（最后20 %，四舍五入到至少一行）。结果数据集在训练集中有3，564，579 个字符，在测试集中有870，014 个字符。这些数据基本上是不平衡的，许多角色只有几行，而少数角色有大量的行。此外，观察测试集不是线的随机样本，而是由每个播放的时间顺序在时间上分开的。使用相同的训练/测试分割，我们还形成了数据集的平衡和IID版本，也有1146 个客户端。

在这些数据上，我们训练了一个堆叠的字符级LSTM语言模型，它在阅读一行中的每个字符后，预测下一个字符[22]。该模型将一系列字符作为输入，并将每个字符嵌入到学习的8维空间中。然后，嵌入的字符通过2个LSTM层进行处理，每个层有256 个节点。最后，第二个LSTM层的输出被发送到softmax 输出层，每个字符一个节点。完整的模型有866，578 个参数，我们使用80 个字符的展开长度进行训练。

SGD对学习率参数η的调整敏感。这里报告的结果是基于在足够宽的学习率网格上进行的训练（在分辨率为10 或10 的乘法网格上，η通常为11 - 13 个值）。我们检查以确保最佳学习率位于网格的中间，并且最佳学习率之间没有显著差异。除非另有说明，否则我们为每个x轴值单独选择的最佳执行率绘制指标。我们发现，最佳的学习率不会变化太多的其他参数的函数。

增加并行性我们首先使用客户端分数C进行实验，它控制多客户端并行性的数量。表1显示了两种MNIST 模型的不同C的影响。我们报告的通信轮数需要达到目标测试集的准确性。为了计算这一点，我们为每个参数设置组合构建了一条学习曲线，如上所述优化η，然后通过采用测试集准确度的最佳值来使每条曲线单调改进，

5我们对这些数据集的不平衡版本进行了额外的实验，发现它们实际上对FedAvg 来说稍微容易一些。

我们总是用character 来表示一个字节的字符串，用role 来表示剧中的一个角色。

Table 1: Effect of the client fraction C on the MNIST 2NN with $E = 1$ and CNN with $E = 5$. Note $C = 0.0$ corresponds to one client per round; since we use 100 clients for the MNIST data, the rows correspond to 1, 10 20, 50, and 100 clients. Each table entry gives the number of rounds of communication necessary to achieve a test-set accuracy of 97% for the 2NN and 99% for the CNN, along with the speedup relative to the $C = 0$ baseline. Five runs with the large batch size did not reach the target accuracy in the allowed time.

2NN	IID		NON-IID	
C	$B = \infty$	$B = 10$	$B = \infty$	$B = 10$
0.0	1455	316	4278	3275
0.1	1474 (1.0×)	87 (3.6×)	1796 (2.4×)	664 (4.9×)
0.2	1658 (0.9×)	77 (4.1×)	1528 (2.8×)	619 (5.3×)
0.5	— (—)	75 (4.2×)	— (—)	443 (7.4×)
1.0	— (—)	70 (4.5×)	— (—)	380 (8.6×)
CNN, $E = 5$				
0.0	387	50	1181	956
0.1	339 (1.1×)	18 (2.8×)	1100 (1.1×)	206 (4.6×)
0.2	337 (1.1×)	18 (2.8×)	978 (1.2×)	200 (4.8×)
0.5	164 (2.4×)	18 (2.8×)	1067 (1.1×)	261 (3.7×)
1.0	246 (1.6×)	16 (3.1×)	— (—)	97 (9.9×)

all prior rounds. We then calculate the number of rounds where the curve crosses the target accuracy, using linear interpolation 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 $B = \infty$ (for MNIST processing all 600 client examples as a single batch per round), there is only a small advantage in increasing the client fraction. Using the smaller batch size $B = 10$ shows a significant improvement in using $C \geq 0.1$, especially in the non-IID case. Based on these results, for most of the remainder of our experiments we fix $C = 0.1$, which strikes a good balance between computational efficiency and convergence rate. Comparing the number of rounds for the $B = \infty$ and $B = 10$ columns in Table 1 shows a dramatic speedup, which we investigate next.

Increasing computation per client In this section, we fix $C = 0.1$, and add more computation per client on each round, either decreasing B, increasing E, or both. Figure 2 demonstrates that adding more local SGD updates per round can produce a dramatic decrease in communication costs, and Table 2 quantifies these speedups. The expected number of updates per client per round is $u = (\mathbb{E}[n_k]/B)E = nE/(KB)$, where the expectation is over the draw of a random client k. We order the rows in each section of Table 2 by this statistic. We see that increasing u by varying both E and B is effective. As long as B is large enough to take full advantage of available parallelism on the client hardware, there is essentially no cost in computation time for lowering it, and so in practice this should be the first parameter tuned.

Table 2: Number of communication rounds to reach a target accuracy for `FedAvg`, versus `FedSGD` (first row, $E = 1$ and $B = \infty$). The u column gives $u = En/(KB)$, the expected number of updates per round.

MNIST CNN, 99% ACCURACY					
CNN	E	B	u	IID	NON-IID
FEDSGD	1	∞	1	626	483
FEDAVG	5	∞	5	179 (3.5×)	1000 (0.5×)
FEDAVG	1	50	12	65 (9.6×)	600 (0.8×)
FEDAVG	20	∞	20	234 (2.7×)	672 (0.7×)
FEDAVG	1	10	60	34 (18.4×)	350 (1.4×)
FEDAVG	5	50	60	29 (21.6×)	334 (1.4×)
FEDAVG	20	50	240	32 (19.6×)	426 (1.1×)
FEDAVG	5	10	300	20 (31.3×)	229 (2.1×)
FEDAVG	20	10	1200	18 (34.8×)	173 (2.8×)
SHAKESPEARE LSTM, 54% ACCURACY					
LSTM	E	B	u	IID	NON-IID
FEDSGD	1	∞	1.0	2488	3906
FEDAVG	1	50	1.5	1635 (1.5×)	549 (7.1×)
FEDAVG	5	∞	5.0	613 (4.1×)	597 (6.5×)
FEDAVG	1	10	7.4	460 (5.4×)	164 (23.8×)
FEDAVG	5	50	7.4	401 (6.2×)	152 (25.7×)
FEDAVG	5	10	37.1	192 (13.0×)	41 (95.3×)

For the IID partition of the MNIST data, using more computation per client decreases the number of rounds to reach the target accuracy by $35\times$ for the CNN and $46\times$ for the 2NN (see Table 4 in Appendix A for details for the 2NN). The speedups for the pathologically partitioned non-IID data are smaller, but still substantial (2.8 – $3.7\times$). It is impressive that averaging provides *any* advantage (vs. actually diverging) when we naively average the parameters of models trained on entirely different pairs of digits. Thus, we view this 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 kind of data distribution we expect for real-world applications. Encouragingly, for this problem learning on the non-IID and unbalanced data is actually much easier (a $95\times$ speedup vs $13\times$ for the balanced IID data); we conjecture this is largely due 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 higher level of test-set accuracy than the baseline `FedSGD` models. This trend continues even if the lines are extended beyond the plotted ranges. For example, for the CNN the $B = \infty, E = 1$ `FedSGD` model eventually reaches 99.22% accuracy after 1200 rounds (and had not improved further after 6000 rounds), while the $B = 10, E = 20$ `FedAvg` model reaches an accuracy of 99.44% after 300 rounds. We conjecture that in addition to lowering communication costs, model averaging produces a regularization benefit similar to that achieved by dropout [36].

We are primarily concerned with generalization performance, but `FedAvg` is effective at optimizing the training loss as well, even beyond the point where test-set accuracy plateaus. We observed similar behavior for all three model classes, and present plots for the MNIST CNN in Figure 6

表1：客户端分数C对E = 1的MNIST 2NN 和E = 5的CNN 的影响。注意C = 0.0 对应于每轮一个客户端;由于我们使用100 个客户端作为MNIST 数据，因此行对应于1、10、20、50 和100 个客户端。每个表条目给出了实现2NN 的97 %测试集准确度和CNN 的99 %测试集准确度所需的通信轮数，沿着相对于C = 0基线的加速。大批量的5次运行在允许的时间内未达到目标准确度。

2、N-Iid

C B = ∞ B = 10 B = ∞ B = 10

0.0 1455 316 4278 3275

0.1 1474 (1.0×) 87 (3.6×) 1796 (2.4×) 664 (4.9×)

0.5 - (-) 75 (4.2 x) - (-) 443 (7.4 x)

CNN， E = 5

0.0 387 50 1181 956

0.1 339 (1.1 x) 18 (2.8 x) 1100 (1.1 x) 206 (4.6 x)

0.2 337 (1.1 x) 18 (2.8 x) 978 (1.2 x) 200 (4.8 x)

0.5 164 (2.4 x) 18 (2.8 x) 1067 (1.1 x) 261 (3.7 x)

之前的所有回合然后，我们使用形成曲线的离散点之间的线性插值来计算曲线与目标精度相交的轮次数。参考图2可能最好地理解这一点，其中灰色线显示了目标。

当B = ∞时（对于MNIST，每轮处理所有600 个客户端示例作为一个批处理），增加客户端比例只有很小的优势。使用较小批量B = 10 显示使用C ≥ 0.1 时显著改善，尤其是在非IID情况下。基于这些结果，对于我们实验的大部分剩余部分，我们固定C = 0.1，这在计算效率和收敛速度之间取得了良好的平衡。比较表1 中B = ∞和B = 10 列的轮数，可以看到显著的加速，我们接下来将对此进行研究。

增加每个客户端的计算量在本节中，我们将C = 0. 1固定下来，并在每轮增加每个客户端的计算量，减少B，增加E，或两者兼而有之。图2表明，每轮增加更多的本地SGD 更新可以显著降低通信成本，表2量化了这些加速。每轮每个客户端的更新的预期数量是u = （E[n]/ B）E = nE/ （KB），其中期望超过随机客户端k的抽取。我们根据这个统计量对表2 中每个部分的行进行排序。我们看到通过改变E和B来增加u是有效的。只要B足够大，可以充分利用客户端硬件上的可用并行性，降低它基本上不会花费计算时间，因此在实践中，这应该是第一个调整的参数。

表2：达到FedAvg 目标准确度的通信回合数与FedSGD （第一行，E = 1且B = ∞）。u列给出u = En/ （KB），即每轮更新的预期数量。

CNN 99%

CNN E B u IID N—IID网络

FSGD 1 ≈ 1 626 483

FA5 ∞ 5 179 (3.5×) 1000 (0.5×) FA1 50 12 65 (9.6×) 600 (0.8×) FA20 ∞ 20 234 (2.7×) 672 (0.7×) FA1 10 60 34 (18.4×) 350 (1.4×) FA5 50 60 29 (21.6×) 334 (1.4×) FA20 50 240 32 (19.6×) 426 (1.1×) FA5 10 300 20 (31.3×) 229 (2.1×) FA20 10 1200 18 (34.8×) 173 (2.8×)

重试 错误原因

SMST，54%

LSTM E B u IID N—IID

FSGD 1 ∞ 1.0 2488 3906 FA 1 50 1.5 1635 （1.5 ×）549 （7.1 ×）FA 5 ∞ 5.0 613 （4.1 ×）597 （6.5 ×）FA 1 10 7.4 460 （5.4 ×）164 （23.8 ×）FA 5 50 7.4 401 （6.2 ×）152 （25.7 ×）FA 5 10 37.1 192 （13.0 ×）4.1 （5.1 × （5.3 ×）

对于MNIST 数据的IID分区，每个客户端使用更多的计算将达到目标精度的轮次数减少了35 倍（对于CNN ）和46 倍（对于2NN ）（参见附录A中的表4了解2NN 的详细信息）。病理分区的非IID数据的加速比较小，但仍然很大（2.8 - 3.7 倍）。令人印象深刻的是，当我们天真地对在完全不同的数字对上训练的模型的参数进行平均时，平均化提供了任何优势（与实际发散相比）。因此，我们认为这是该方法稳健性的有力证据。

莎士比亚的不平衡和非IID分布（按戏剧中的角色）更能代表我们对现实世界应用程序所期望的数据分布。令人鼓舞的是，对于这个问题，在非IID和不平衡数据上学习实际上要容易得多（95 倍的加速比平衡IID数据的13 倍）;我们推测这主要是由于某些角色具有相对较大的本地数据集，这使得增加本地训练特别有价值。

对于所有三个模型类，FedAvg 收敛到比基准FedSGD 模型更高的测试集准确度。即使线条延伸到绘制的范围之外，这种趋势也会继续。例如，对于CNN ，B = ∞，E = 1 FedSGD 模型在1200 轮后最终达到99.22 %的准确度（并且在6000 轮后没有进一步提高），而B = 10 ，E = 20 FedAvg 模型在300 轮后达到99.44 %的准确度。我们推测，除了降低通信成本外，模型平均还产生了与dropout 类似的正则化好处[36]。

我们主要关注的是泛化性能，但FedAvg 在优化训练损失方面也很有效，甚至超过了测试集准确度的平台。我们观察到所有三个模型类的类似行为，并在图6中展示了MNIST CNN 的图

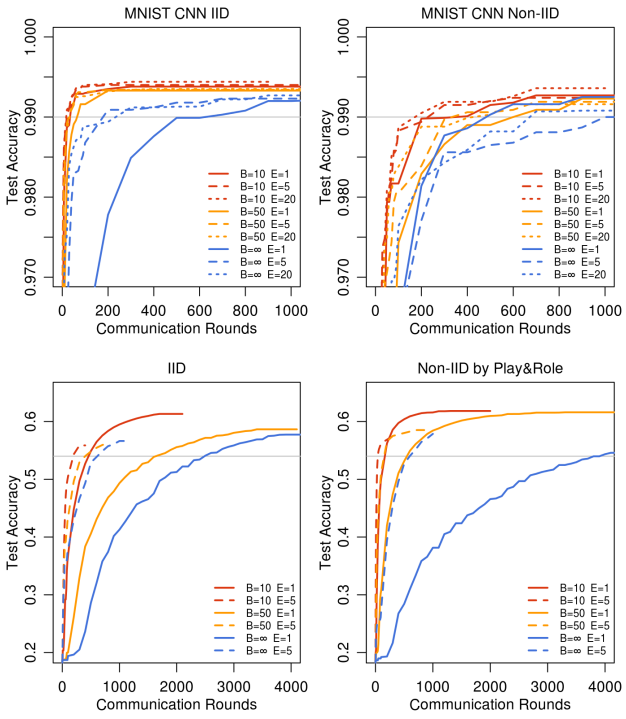


Figure 2: Test set accuracy vs. communication rounds for the MNIST CNN (IID and then pathological non-IID) and Shakespeare LSTM (IID and then by Play&Role) with $C = 0.1$ and optimized η. The gray lines show the target accuracies used in Table 2. Plots for the 2NN are given as Figure 7 in Appendix A.

in Appendix A.

Can we over-optimize on the client datasets? The current model parameters only influence the optimization performed in each `ClientUpdate` via initialization. Thus, as $E \to \infty$, at least for a convex problem eventually the initial conditions should be irrelevant, and the global minimum would be reached regardless of initialization. Even for a non-convex problem, one might conjecture the algorithm would converge to the same local minimum as long as the initialization was in the same basin. That is, we would expect that while one round of averaging might produce a reasonable model, additional rounds of communication (and averaging) would not produce further improvements.

Figure 3 shows the impact of large E during initial training on the Shakespeare LSTM problem. Indeed, for very large numbers of local epochs, `FedAvg` can plateau or diverge.[7] This result suggests that for some models, especially in the later stages of convergence, it may be useful to decay the

[7]Note that due to this behavior and because for large E not all experiments for all learning rates were run for the full number of rounds, we report results for a fixed learning rate (which perhaps surprisingly was near-optimal across the range of E parameters) and without forcing the lines to be monotonic.

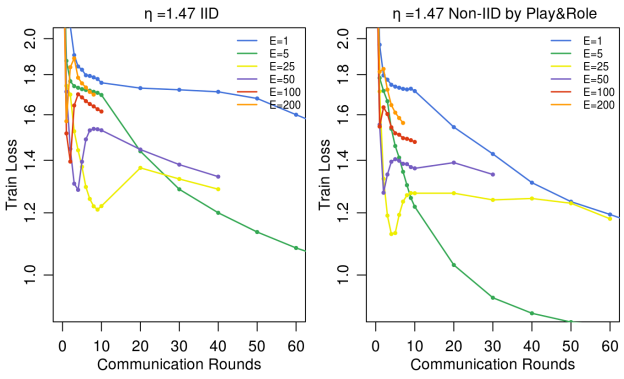


Figure 3: The effect of training for many local epochs (large E) between averaging steps, fixing $B = 10$ and $C = 0.1$ for the Shakespeare LSTM with a fixed learning rate $\eta = 1.47$.

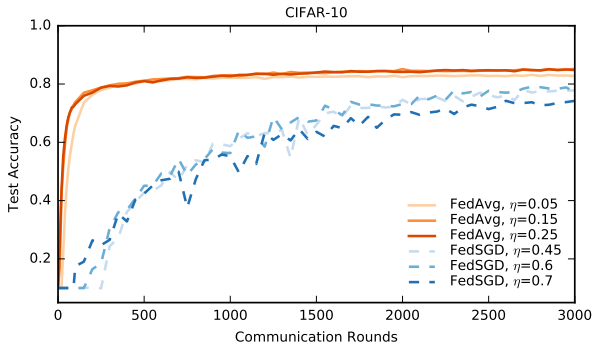
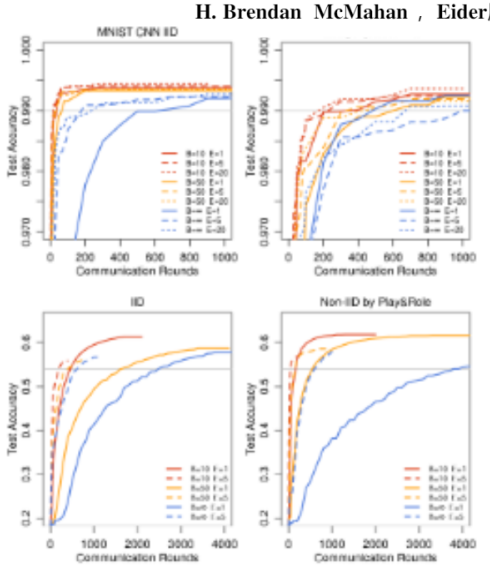


Figure 4: Test accuracy versus communication for the CIFAR10 experiments. `FedSGD` uses a learning-rate decay of 0.9934 per round; `FedAvg` uses $B = 50$, learning-rate decay of 0.99 per round, and $E = 5$.

amount of local computation per round (moving to smaller E or larger B) in the same way decaying learning rates can be useful. Figure 8 in Appendix A gives the analogous experiment for the MNIST CNN. Interestingly, for this model we see no significant degradation in the convergence rate for large values of E. However, we see slightly better performance for $E = 1$ versus $E = 5$ for the large-scale language modeling task described below (see Figure 10 in Appendix A).

CIFAR experiments We also ran experiments on the CIFAR-10 dataset [24] to further validate `FedAvg`. The dataset consists of 10 classes of 32x32 images with three RGB channels. There are 50,000 training examples and 10,000 testing examples, which we partitioned into 100 clients each containing 500 training and 100 testing examples; since there isn't a natural user partitioning of this data, we considered the balanced and IID setting. The model architecture was taken from the TensorFlow tutorial [38], which consists of two convolutional layers followed by two fully connected layers and then a linear transformation layer



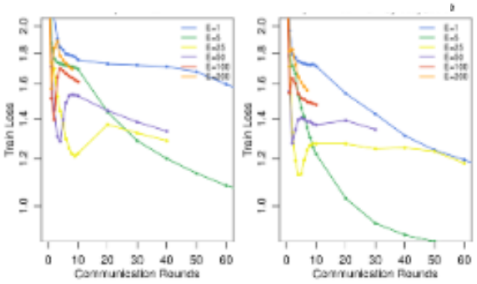


图2：测试集准确度与MNIST CNN （IID，然后是病理性非IID）和Shakespeare LSTM（IID，然后是Play＆Role ）的通信回合，C = 0.1 ，优化η。灰色线显示了表2中使用的目标精度。2NN 的曲线如附录A中的图7所示。

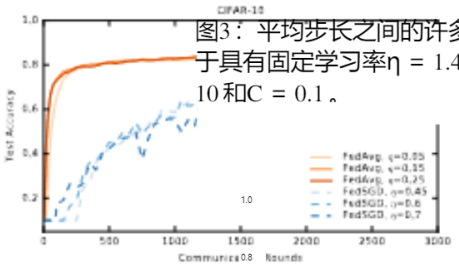


图3：平均步长之间的许多局部时期（大E）的训练效果，对于具有固定学习率η = 1.47 的Shakespeare LSTM，固定B = 10 和C = 0.1 。

图4：CIFAR 10 实验的测试准确度与通信。FedSGD 使用每轮0.9934 的学习率衰减; FedAvg 使用B = 50 ，每轮0.99 的学习率衰减，E = 5。

附录A中

我们可以在客户端数据集上过度优化吗？当前模型参数仅影响通过初始化在每个ClientUpdate 中执行的优化。因此，当E → ∞时，至少对于一个凸问题，初始条件最终应该是无关的，并且无论初始化如何，都将达到全局最小值。即使对于非凸问题，只要初始化在同一流域内，人们也可以推测算法将收敛到相同的局部最小值。也就是说，我们希望一轮平均可能会产生一个合理的模型，额外的几轮沟通（和平均）不会产生进一步的改进。

图3显示了初始训练期间大E对Shakespeare LSTM问题的影响。事实上，对于大量的局部epoch ，FedAvg 可以达到平台或发散。这一结果表明，对于某些模型，特别是在收敛的后期阶段，

[7]请注意，由于这种行为，并且由于对于大E，并非所有学习率的所有实验都运行了完整的轮数，我们报告了固定学习率的结果（这可能令人惊讶地在E参数范围内接近最佳），并且没有强制线单调。

每轮的局部计算量（移动到更小的E或更大的B）以相同的方式衰减学习速率可能是有用的。附录A中的图8给出了MNIST CNN 的类似实验。有趣的是，对于这个模型，我们没有看到大的E值的收敛速度显着下降。然而，对于下面描述的大规模语言建模任务，我们看到E = 1比E = 5的性能稍好（参见附录A中的图10 ）。

CIFAR实验我们还在CIFAR-10 数据集上进行了实验[24]，以进一步验证FedAvg 。该数据集由10 类32 x32 图像组成，具有三个RGB通道。有50，000 个训练示例和10，000 个测试示例，我们将其划分为100 个客户端，每个客户端包含500 个训练示例和100 个测试示例;由于这些数据没有自然的用户分区，因此我们考虑了平衡和IID设置。模型架构取自TensorFlow 教程[38]，它包括两个卷积层，然后是两个全连接层，然后是线性变换层

Table 3: Number of rounds and speedup relative to baseline SGD to reach a target test-set accuracy on CIFAR10. SGD used a minibatch size of 100. FedSGD and FedAvg used $C = 0.1$, with FedAvg using $E = 5$ and $B = 50$.

ACC.	80%	82%	85%
SGD	18000 (—)	31000 (—)	99000 (—)
FEDSGD	3750 (4.8×)	6600 (4.7×)	N/A (—)
FEDAVG	280 (64.3×)	630 (49.2×)	2000 (49.5×)

to produce logits, for a total of about 10^6 parameters. Note that state-of-the-art approaches have achieved a test accuracy of 96.5% [19] for CIFAR; nevertheless, the standard model we use is sufficient for our needs, as our goal is to evaluate our optimization method, not achieve the best possible accuracy on this task. The images are preprocessed as part of the training input pipeline, which consists of cropping the images to 24x24, randomly flipping left-right and adjusting the contrast, brightness and whitening.

For these experiments, we considered an additional baseline, standard SGD training on the full training set (no user partitioning), using minibatches of size 100. We achieved an 86% test accuracy after 197,500 minibatch updates (each minibatch update requires a communication round in the federated setting). FedAvg achieves a similar test accuracy of 85% after only 2,000 communication rounds. For all algorithms, we tuned a learning-rate decay parameter in addition to the initial learning rate. Table 3 gives the number of communication rounds for baseline SGD, FedSGD, and FedAvg to reach three different accuracy targets, and Figure 4 gives learning-rate curves for FedAvg versus FedSGD.

By running experiments with minibatches of size $B = 50$ for both SGD and FedAvg, we can also look at accuracy as a function of the number of such minibatch gradient calculations. We expect SGD to do better here, because a sequential step is taken after each minibatch computation. However, as Figure 9 in the appendix shows, for modest values of C and E, FedAvg makes a similar amount of progress per minibatch computation. Further, we see that both standard SGD and FedAvg with only one client per round ($C = 0$), demonstrate significant oscillations in accuracy, whereas averaging over more clients smooths this out.

Large-scale LSTM experiments We ran experiments on a large-scale next-word prediction task to demonstrate the effectiveness of our approach on a real-world problem. Our training dataset consists 10 million public posts from a large social network. We grouped the posts by author, for a total of over 500,000 clients. This dataset is a realistic proxy for the type of text entry data that would be present on a user's mobile device. We limited each client dataset to at most 5000 words, and report accuracy (the fraction of the data where the highest predicted probability was on the correct next word, out of 10000 possibilities) on a test set of $1e5$

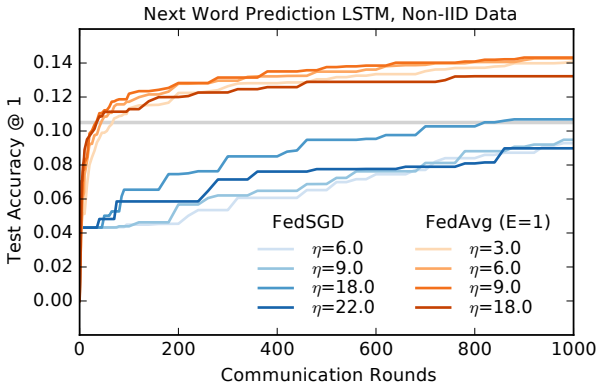


Figure 5: Monotonic learning curves for the large-scale language model word LSTM.

posts from different (non-training) authors. Our model is a 256 node LSTM on a vocabulary of 10,000 words. The input and output embeddings for each word were of dimension 192, and co-trained with the model; there are 4,950,544 parameters in all. We used an unroll of 10 words.

These experiments required significant computational resources and so we did not explore hyper-parameters as thoroughly: all runs trained on 200 clients per round; FedAvg used $B = 8$ and $E = 1$. We explored a variety of learning rates for FedAvg and the baseline FedSGD. Figure 5 shows monotonic learning curves for the best learning rates. FedSGD with $\eta = 18.0$ required 820 rounds to reach 10.5% accuracy, while FedAvg with $\eta = 9.0$ reached an accuracy of 10.5% in only 35 communication rounds (23× fewer then FedSGD). We observed lower variance in test accuracy for FedAvg, see Figure 10 in Appendix A. This figure also include results for $E = 5$, which performed slightly worse than $E = 1$.

4 Conclusions and Future Work

Our experiments show that federated learning can be made practical, as FedAvg trains high-quality models using relatively few rounds of communication, as demonstrated by results on a variety of model architectures: a multi-layer perceptron, two different convolutional NNs, a two-layer character LSTM, and a large-scale word-level LSTM.

While federated learning offers many practical privacy benefits, providing stronger guarantees via differential privacy [14, 13, 1], secure multi-party computation [18], or their combination is an interesting direction for future work. Note that both classes of techniques apply most naturally to synchronous algorithms like FedAvg.[8]

[8] Subsequent to this work, Bonawitz et al. [6] introduced an efficient secure aggregation protocol for federated learning, and Konečný et al. [23] presented algorithms for further decreasing communication costs.

其包括两个卷积层，接着是两个全连接层，然后是线性变换层表3：在CIFAR 10 上达到目标测试集精度的相对于基线SGD的轮数和加速。SGD使用的小批量为100 。FedSGD 和FedAvg 使用C = 0.1 ，FedAvg 使用E = 5和B = 50 。

a. 80 % 82 % 85 % SGD 18000 （—） 31000 （—） 99000 （—） FSGD 3750 （4.8 ×） 6600 （4.7 ×） / （—） FA 280 （64.3 ×） 630 （49.2 ×） 2000 （49.5 ×） 2000 （49.5 ×）

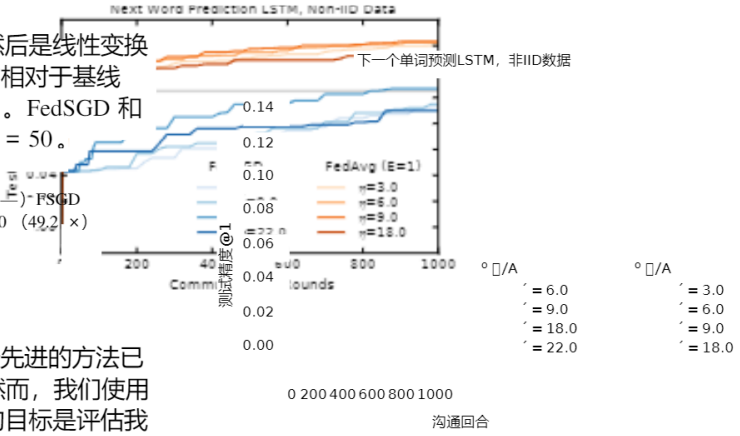


图5：大规模语言模型词LSTM的单调学习曲线。

以产生logits ，总共约10 个参数。请注意，最先进的方法已经实现了CIFAR的96.5 %的测试准确度[19];然而，我们使用的标准模型足以满足我们的需求，因为我们的目标是评估我们的优化方法，而不是在这项任务上实现最佳的准确度。图像作为训练输入管道的一部分进行预处理，包括将图像裁剪为24 x24 ，随机左右翻转以及调整对比度，亮度和白化。

对于这些实验，我们考虑了一个额外的基线，在完整的训练集上进行标准SGD训练（没有用户分区），使用大小为100 的小批量。在197，500 次小批量更新之后，我们实现了86 %的测试准确率（每个小批量更新都需要在联邦设置中进行一轮通信）。FedAvg 在仅仅2,000 轮通信之后就达到了85 %的类似测试准确率。对于所有算法，除了初始学习率之外，我们还调整了学习率衰减参数。表3给出了基准SGD、FedSGD 和FedAvg 达到三个不同精度目标所需的通信回合数，图4给出了FedAvg 与FedSGD 的学习率曲线。

通过对SGD和FedAvg 运行大小为B = 50 的小批量的实验，我们还可以查看精度作为这种小批量梯度计算的数量的函数。我们希望SGD在这里做得更好，因为在每个小批量计算之后都要执行一个顺序步骤。然而，如附录中的图9所示，对于适当的C和E值，FedAvg 在每个小批处理计算中取得了类似的进度。此外，我们看到，标准SGD和FedAvg 每轮只有一个客户端（C = 0），在准确性方面表现出显着的波动，而对更多客户端进行平均则会平滑这一点。

大规模LSTM实验我们在大规模下一个单词预测任务上进行了实验，以证明我们的方法在现实世界问题上的有效性。我们的训练数据集由来自大型社交网络的1000 万个公共帖子组成。我们按作者对帖子进行了分组，总共有50 多万客户。该数据集是将存在于用户的移动终端上的文本输入数据类型的现实代理。我们将每个客户端数据集限制为最多5000 个单词，并在1e5 的测试集上报告准确性（在10000 个可能性中，下一个单词的预测概率最高的数据部分

来自不同（非培训）作者的帖子。我们的模型是一个256 节点的LSTM，词汇量为10，000 个单词。每个单词的输入和输出嵌入的维度为192 ，并与模型共同训练;总共有4，950，544 个参数。我们用了10 个字展开。

这些实验需要大量的计算资源，因此我们没有彻底探索超参数：所有运行都在每轮200 个客户端上训练; FedAvg 使用B = 8和E = 1。我们探索了FedAvg 和基准FedSGD 的各种学习率。图5显示了最佳学习率的单调学习曲线。η = 18.0 的FedSGD 需要820 轮才能达到10.5 %的准确度，而η = 9.0 的FedAvg 只需要35 轮就能达到10.5 %的准确度（比FedSGD 少23 倍）。我们观察到FedAvg 的测试准确度方差较低，参见附录A中的图10 。该图还包括E = 5的结果，其表现略差于E = 1 。

4结论和未来工作

我们的实验表明，联邦学习可以变得实用，因为FedAvg 使用相对较少的通信轮次来训练高质量的模型，正如各种模型架构的结果所证明的那样：多层感知器，两种不同的卷积NN ，两层字符LSTM和大规模单词级LSTM。

虽然联邦学习提供了许多实际的隐私优势，但通过差分隐私[14，13，1]，安全多方计算[18]或其组合提供更强的保证是未来工作的一个有趣方向。请注意，这两类技术最自然地适用于FedAvg 等同步算法。

在这项工作之后，Bonawitz 等人[6]为联邦学习引入了一种有效的安全聚合协议，而Kone Escn'y 等人[23]提出了进一步降低通信成本的算法。

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *23rd ACM Conference on Computer and Communications Security (ACM CCS)*, 2016.

[2] Monica Anderson. Technology device ownership: 2015. http://www.pewinternet.org/2015/10/29/technology-device-ownership-2015/, 2015.

[3] Yossi Arjevani and Ohad Shamir. Communication complexity of distributed convex learning and optimization. In *Advances in Neural Information Processing Systems 28*. 2015.

[4] Maria-Florina Balcan, Avrim Blum, Shai Fine, and Yishay Mansour. Distributed learning, communication complexity and privacy. *arXiv preprint arXiv:1204.3514*, 2012.

[5] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 2003.

[6] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for federated learning on user-held data. In *NIPS Workshop on Private Multi-Party Machine Learning*, 2016.

[7] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2), 1981.

[8] Jianmin Chen, Rajat Monga, Samy Bengio, and Rafal Jozefowicz. Revisiting distributed synchronous sgd. In *ICLR Workshop Track*, 2016.

[9] Anna Choromanska, Mikael Henaff, Michaël Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *AISTATS*, 2015.

[10] Greg Corrado. Computer, respond to this email. http://googleresearch.blogspot.com/2015/11/computer-respond-to-this-email.html, November 2015.

[11] Yann N. Dauphin, Razvan Pascanu, Çaglar Gülçehre, KyungHyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *NIPS*, 2014.

[12] Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, Marc'Aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, and Andrew Y. Ng. Large scale distributed deep networks. In *NIPS*, 2012.

[13] John Duchi, Michael I. Jordan, and Martin J. Wainwright. Privacy aware learning. *Journal of the Association for Computing Machinery*, 2014.

[14] Cynthia Dwork and Aaron Roth. *The Algorithmic Foundations of Differential Privacy*. Foundations and Trends in Theoretical Computer Science. Now Publishers, 2014.

[15] Olivier Fercoq, Zheng Qu, Peter Richtárik, and Martin Takác. Fast distributed coordinate descent for non-strongly convex losses. In *Machine Learning for Signal Processing (MLSP), 2014 IEEE International Workshop 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 optimization problems. In *ICLR*, 2015.

[18] Slawomir Goryczka, Li Xiong, and Vaidy Sunderam. Secure multiparty aggregation with differential privacy: A comparative study. In *Proceedings of the Joint 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 Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8), November 1997.

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[22] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. Character-aware neural language models. *CoRR*, abs/1508.06615, 2015.

[23] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtarik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. In *NIPS Workshop on Private Multi-Party Machine Learning*, 2016.

[24] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*. 2012.

[26] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 1998.

[27] Chenxin Ma, Virginia Smith, Martin Jaggi, Michael I Jordan, Peter Richtárik, and Martin Takáč. Adding vs. averaging in distributed primal-dual optimization. In *ICML*, 2015.

引用

[1]Martin Abadi 、Andy Chu 、Ian Goodfellow 、Brendan McMahan 、Ilya Mironov 、Kunal Talwar 和Li Zhan深度学习与差分隐私第23 届ACM 计算机与通信安全会议（*ACM CCS*），*2016* 年。

[2]莫妮卡安德森。技术设备所有权：2015 年。http ：//www.pewinternet.org/2015/10/29/ technology-device ownership-2015/，2015.

[3]尤西·阿杰瓦尼和奥哈德·沙米尔分布式凸学习和优化的通信复杂性 异化。在神经信息处理系统的进展*28* 。*2015* .

[4]Maria -Florina Balcan ，Avrim Blum ，Shai Fine ，and Yishay Mansour .分布式学习、通信复杂性和隐私。arXiv预印本arXiv：1204.3514，2012 年。

[5]约瑟芬·本吉奥、雷让·杜夏姆、帕斯卡·文森特和克里斯蒂安·扬文。神经概率语言模型。J·马赫。学习.结果：2003 . [6]放大图片作者：基思博纳维茨，弗拉基米尔伊万诺夫，本克罗伊特，安东尼奥Marcedone ，H.布兰登·麦克马汉，萨瓦尔·帕特尔，丹尼尔·拉米奇，亚伦·西格尔，卡恩·赛斯。用户持有的联邦学习的实用安全聚合 数据在*NIPS* 关于私人多方机器学习的研讨会上，*2016* 年。

[7]大卫湖乔姆无法追踪的电子邮件，回邮地址和数字缩写。Commun . ACM，24 （2），1981 年。[8]Jianmin Chen ，Rajat Monga ，Samy Bengio ，and Rafal Jozefowicz .再谈分布式同步sgd 。 在*ICLR*研讨会轨道，*2016* 年。

[9]安娜·乔罗曼斯卡、米克尔·赫纳夫、米夏埃尔·马蒂厄、杰勒德·本·阿鲁斯和扬·勒昆。多层网络的损耗面。In AISTATS，2015 .

[10]格雷格·科拉多电脑，重新-spond到这个电子邮件。联系我们 googleresearch.blogspot.com/2015/ 11/computer respond-to-this-email .html 》，2015 年11 月。

[11]扬恩·N。Dauphin 、Razvan Pascanu 、C á aglar g¨ul c kangehre 、KyungHyun Cho 、Surya Ganguli 和Yoshua Bengi高维非凸优化问题鞍点问题的识别与求解。在NIPS ，2014 年。

[12]作者：Jeffrey Dean ，Greg S. Corrado ，Rajat Monga ，Kai 陈，Matthieu Devin ，Quoc V. Le，Mark Z。Mao ，Marc 'Aurelio Ranzato ，Andrew Senior ，Paul Tucker ，Ke Yang ，and Andrew Y。吴。大型分布式深度网络（Large Distributed Deep Networks ）在

[13]约翰·杜奇，迈克尔·I.乔丹和马丁·韦恩-赖特。隐私意识学习。*Journal of the Association for Computing Machinery* ，*2014* .

[14]辛西娅·德沃克和亚伦·罗斯的算法差异隐私的基础。基金会和理论计算机科学的趋势。出版社，2014 年。

[15]Olivier Fercoq ，郑楚，Peter Richt 'arik ，和马丁塔克非强凸损失的快速分布式坐标下降。信号处理机器学习（MLSP），2014 年IEEE国际研讨会，2014 年。

[16]Ian Goodfellow ，Yooney Bengio ，and Aaron Courville . 深度学习本书准备由MIT出版社出版，2016 年。

[17]Ian J. Goodfellow ，Oriol Vinyals 和Andrew M.萨克斯定性描述神经网络优化问题。2015 年，国际会议。

[18]Slawomir Goryczka ，Li Xiong ，and Vaidy Sunderam . 具有差异隐私的安全多方聚合：比较研究。在EDBT/ICDT 2013 年联合研讨会的会议记录中，2013 年。

[19]本杰明·格雷厄姆分数最大池化。CoRR ，abs/1412.6071，2014 网址http://arxivorg/abs/1412.6071

[20]塞普Hochreiter 和J 'urgen Schmidhuber 。Long Short ——术语记忆Neural Computation ，9 （8），November 1997 .

[21]谢尔盖·约菲和克里斯蒂安·塞格迪。批量标准化-·通过减少内部协变量偏移来加速深度网络训练。In ICML，2015 .

[22]Yoon Kim、Yacine Jernite 、大卫桑塔格和亚历山大M.急神经感知语言模型 CoRR ，abs /1508.06615，2015 年。

[23]Jakub Kone po cn“ y” ，H。布伦丹·麦克马汉，费利克斯·X。于、Peter Richtarik ，Ananda Theertha Suresh ，and Dave Bacon .联邦学习：提高沟通效率的策略。在NIPS 关于私人多方机器学习的研讨会上，2016 年。

[24]亚历克斯·克里热夫斯基从微小图像中学习多层特征。技术报告，2009 年。

[25]Alex Krizhevsky ，Ilya Sutskever 和Geoffrey E.欣... 吨使用深度卷积神经网络进行图像网分类。在NIPS 。2012 .

[26]Y.勒昆湖博图湾，巴西-地Bengio 和P. Haffner 的研究。应用于文档识别的基于一致性的学习。Proceedings of the IEEE, 86 （11），1998 .

[27]马晨欣、弗吉尼亚·史密斯、马丁·贾吉、迈克尔·I Jordan ，Peter Richt 'arik ，and Martin Tak' a Brachic .分布式原始对偶优化中的加法与平均。In ICML，2015 .

[28] Ryan McDonald, Keith Hall, and Gideon Mann. Distributed training strategies for the structured perceptron. In *NAACL HLT*, 2010.

[29] Natalia Neverova, Christian Wolf, Griffin Lacey, Lex Fridman, Deepak Chandra, Brandon Barbello, and Graham W. Taylor. Learning human identity from motion patterns. *IEEE Access*, 4:1810–1820, 2016.

[30] Jacob Poushter. Smartphone ownership and internet usage continues to climb in emerging economies. Pew Research Center Report, 2016.

[31] Daniel Povey, Xiaohui Zhang, and Sanjeev Khudanpur. Parallel training of deep neural networks with natural gradient and parameter averaging. In *ICLR Workshop Track*, 2015.

[32] William Shakespeare. The Complete Works of William Shakespeare. Publically available at `https://www.gutenberg.org/ebooks/100`.

[33] Ohad Shamir and Nathan Srebro. Distributed stochastic optimization and learning. In *Communication, Control, and Computing (Allerton)*, 2014.

[34] Ohad Shamir, Nathan Srebro, and Tong Zhang. Communication efficient distributed optimization using an approximate newton-type method. *arXiv preprint arXiv:1312.7853*, 2013.

[35] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, 2015.

[36] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. 15, 2014.

[37] Latanya Sweeney. Simple demographics often identify people uniquely. 2000.

[38] TensorFlow team. Tensorflow convolutional neural networks tutorial, 2016. `http://www.tensorflow.org/tutorials/deep_cnn`.

[39] White House Report. Consumer data privacy in a networked world: A framework for protecting privacy and promoting innovation in the global digital economy. *Journal of Privacy and Confidentiality*, 2013.

[40] Tianbao Yang. Trading computation for communication: Distributed stochastic dual coordinate ascent. In *Advances in Neural Information Processing Systems*, 2013.

[41] Ruiliang Zhang and James Kwok. Asynchronous distributed 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.

[43] Yuchen Zhang and Lin Xiao. Communication-efficient distributed optimization of self-concordant empirical loss. *arXiv preprint arXiv:1501.00263*, 2015.

[44] Yuchen Zhang, Martin J Wainwright, and John C Duchi. Communication-efficient algorithms for statistical optimization. In *NIPS*, 2012.

[45] Yuchen Zhang, John Duchi, Michael I Jordan, and Martin J Wainwright. Information-theoretic lower bounds for distributed statistical estimation with communication constraints. In *Advances in Neural Information Processing Systems*, 2013.

[46] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J. Smola. Parallelized stochastic gradient descent. In *NIPS*. 2010.

[28]瑞安·麦克唐纳，基思·霍尔，吉迪恩·曼。这...
结构化感知器的贡献训练策略。在NAACL HLT，2010 年。

[29]娜塔莉亚内韦洛娃，克里斯蒂安沃尔夫，格里芬莱西，莱克斯 Fridman ，Deepak Chandra ，布兰登巴贝罗和 Graham W. Taylor.从运动模式中识别人类身份。IEEE Access ，4：1810 -1820，2016 年。

[30]雅各布·普什特拥有智能手机和互联网
新兴经济体的使用量继续攀升。皮尤研究中心报告，2016 年。

[31]丹尼尔·波维，张晓辉和桑吉夫·库丹普尔。
使用自然梯度和参数平均并行训练深度神经网络。ICLR Workshop Track，2015 年。

[32]威廉·莎士比亚的全集
威廉·莎士比亚可在
https ：//www.gutenberg .org/ebooks/100上公开获取。

[33]奥哈德·沙米尔和内森·斯雷布罗分布式随机-
优化和学习。在通信，控制和计算（*Allerton* ），*2014* 年。

[34]Ohad Shamir Nathan Srebro 和Tong Zhang 。COM -
使用近似牛顿型方法的通信高效分布式优化。arXiv预印本arXiv：1312.7853，2013 年。

[35]雷扎·肖克里和维塔利·什马蒂科夫隐私保护
深度学习第*22* 届*ACM SIGSAC* 计算机和通信安全会议论文集，*CCS '15*，*2015* 。

[36]尼蒂什·斯里瓦斯塔瓦杰弗里·欣顿亚历克斯·克里热夫斯基 Ilya Sutskever 和Ruslan Salakhutdinov Dropout ：防止神经网络过度拟合的简单方法。
15，2014 .

[37]拉坦亚·斯威尼简单的人口统计数据往往能唯一地识别出人们。2000 .

[38]TensorFlow 团队Tensorflow 卷积神经网络-
作品教程，2016 http://www.tensorflow.org/tutorials/deep_cnn.

[39]白宫报告。网络中的消费者数据隐私-
Worked World ：一个保护隐私和促进全球数字经济创新的框架。
隐私和保密杂志，*2013* 年。

[40]杨天宝。交易计算为communica -
分布随机双坐标上升。在
神经信息处理系统的进展，*2013* 。

[41]张瑞良和James Kwok。用于一致性优化的异步分布式admm 。在ICML。
JMLR研讨会和会议记录，2014 年。

[42]Sixin Zhang ，安娜E Choromanska ，和Yann LeCun .
使用弹性平均的深度学习。在NIPS。
2015 .

[43]张雨辰和林晓。沟通效率
自协调经验损失的分布式优化。arXiv预印本arXiv：1501.00263，2015 年。

[44]张宇晨、马丁J温赖特和约翰C
杜奇用于统计优化的通信高效算法。2012 年，在NIPS中。

[45]张宇晨、约翰·杜奇、迈克尔·乔丹和马-
锡J温赖特。分布式统计估计的信息论下界

约束条件。在神经信息处理系统的进展，*2013* 年。

[46]Martin Zinkevich ，Markus Weimer ，Lihong Li，和亚历克斯·J·斯莫拉简化的随机梯度下降。在NIPS。2010 .

A Supplemental Figures and Tables

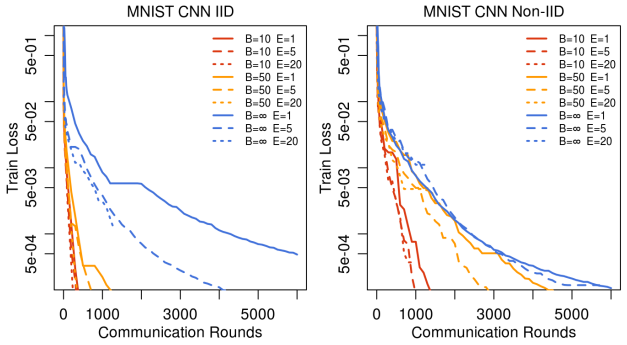


Figure 6: Training set convergence for the MNIST CNN. Note the y-axis is on a log scale, and the x-axis covers more training than Figure 2. These plots fix $C = 0.1$.

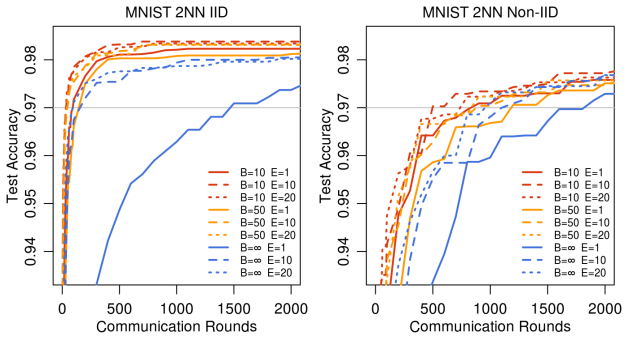


Figure 7: Test set accuracy vs. communication rounds for MNIST 2NN with $C = 0.1$ and optimized η. The left column is the IID dataset, and right is the pathological 2-digits-per-client non-IID data.

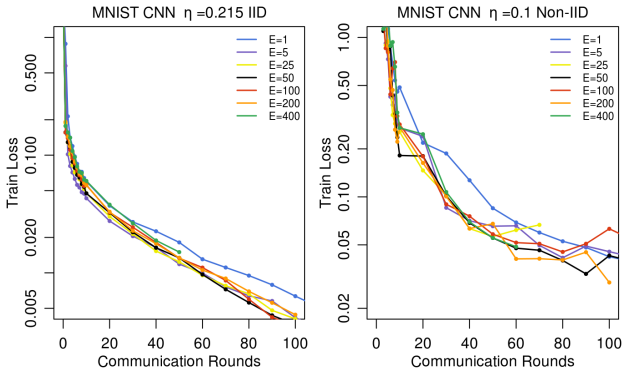


Figure 8: The effect of training for many local epochs (large E) between averaging steps, fixing $B = 10$ and $C = 0.1$. Training loss for the MNIST CNN. Note different learning rates and y-axis scales are used due to the difficulty of our pathological non-IID MNIST dataset.

Table 4: Speedups in the number of communication rounds to reach a target accuracy of 97% for `FedAvg`, versus `FedSGD` (first row) on the MNIST 2NN model.

MNIST 2NN	E	B	u	IID	NON-IID
FEDSGD	1	∞	1	1468	1817
FEDAVG	10	∞	10	156 (9.4×)	1100 (1.7×)
FEDAVG	1	50	12	144 (10.2×)	1183 (1.5×)
FEDAVG	20	∞	20	92 (16.0×)	957 (1.9×)
FEDAVG	1	10	60	92 (16.0×)	831 (2.2×)
FEDAVG	10	50	120	45 (32.6×)	881 (2.1×)
FEDAVG	20	50	240	39 (37.6×)	835 (2.2×)
FEDAVG	10	10	600	34 (43.2×)	497 (3.7×)
FEDAVG	20	10	1200	32 (45.9×)	738 (2.5×)

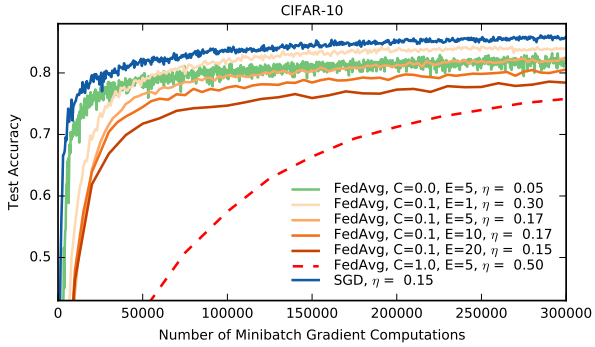


Figure 9: Test accuracy versus number of minibatch gradient computations ($B = 50$). The baseline is standard sequential SGD, as compared to `FedAvg` with different client fractions C (recall $C = 0$ means one client per round), and different numbers of local epochs E.

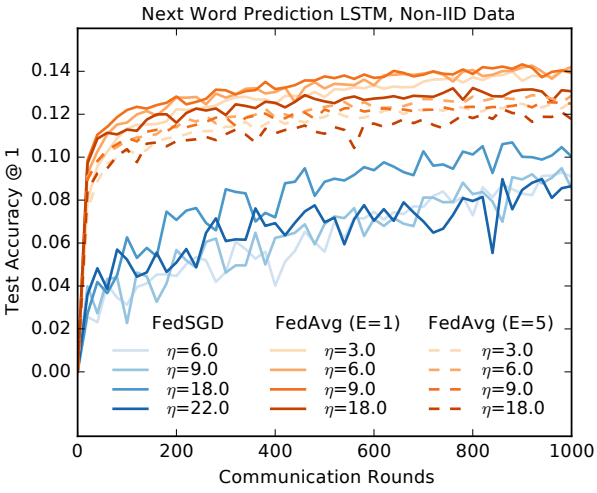


Figure 10: Learning curves for the large-scale language model word LSTM, with evaluation computed every 20 rounds. `FedAvg` actually performs better with fewer local epochs E (1 vs 5), and also has lower variance in accuracy across evaluation rounds compared to `FedSGD`.

A补充图表

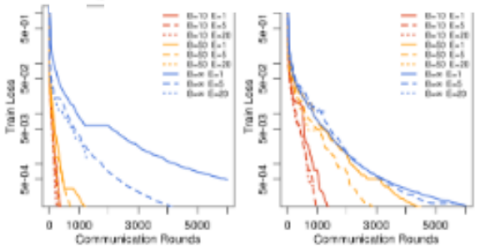


图6： MNIST CNN 的训练集收敛。请注意，y轴是对数刻度，x轴涵盖的训练比图2更多。这些图固定C = 0.1 。

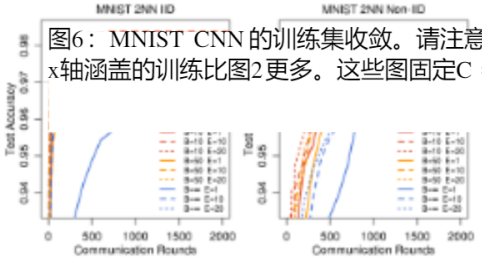


图7： C = 0.1 和优化η的MNIST 2NN 的测试集准确度与通信轮次。左列是IID数据集，右列是病理性每客户端2个数字的非IID数据。

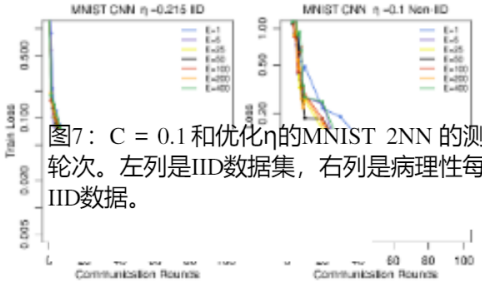
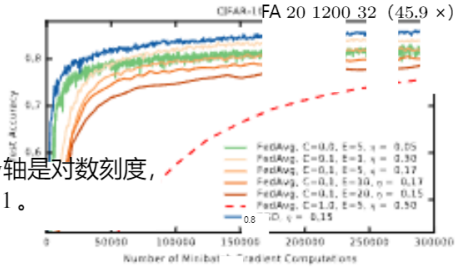


图8： 平均步长之间的许多局部时期（大E）的训练效果，固定B = 10和C = 0.1 。MNIST CNN 的训练损失。注意，由于我们的病理性非IID MNIST数据集的难度，使用了不同的学习率和y轴尺度。

表四： 加快通信轮数，以达到FedAvg 的97 %的目标准确度，与MNIST 2NN 模型上的FedSGD （第一行）相比。

MNIST 2NN	E	B	u	IID	N—IID
FSGD	1	∞	1	1468	1817
FA	10	10	156	（1100 ×）	（1.7 ×）
FA	1	50	12	144	（10.2 ×）
1183 （1.5 ×）					
FA	20	∞	20	92	（16.0 ×）
957 （1.9 ×）					
FA	1	10	60	92	（16.0 × 831）
（2.2 ×）					
FA	10	50	120	45	（32.6 ×）
881 （2.1 ×）					
FA	20	50	240	39	（37.6 ×）
835 （2.2 ×）					
FA	10	600	34	（43.2 ×）	497 （3.7 ×）
FA	20	1200	32	（45.9 ×）	738 （2.5 ×）



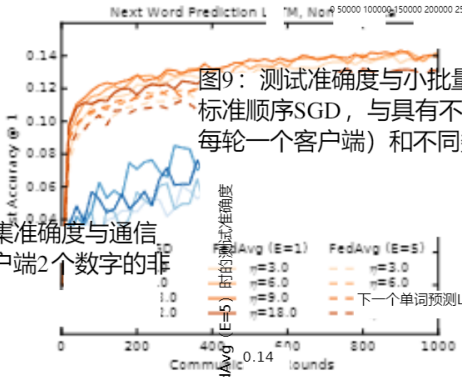


图9： 测试准确度与小批量梯度计算次数（B = 50 ）。基线是标准顺序SGD，与具有不同客户端分数C（召回C = 0意味着每轮一个客户端）和不同数量的本地时期E的FedAvg 相比。

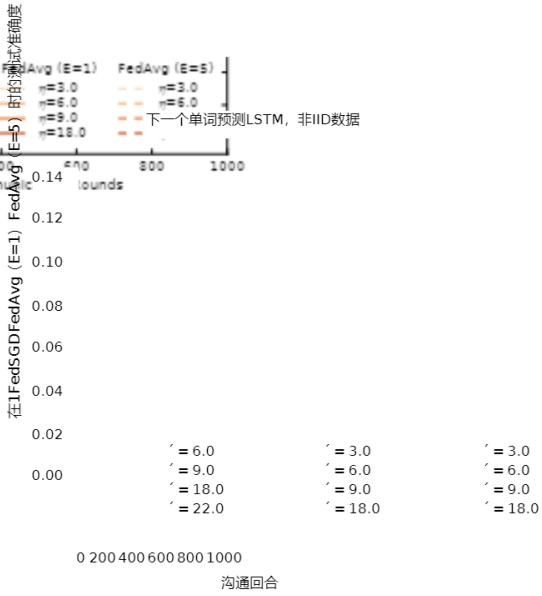


图10： 大规模语言模型单词LSTM的学习曲线，每20 轮计算一次评估。FedAvg 实际上在更少的局部时期E （1 vs 5 ）下表现更好，并且与FedSGD 相比，在评估轮之间的准确性差异也更低。