Communication Efficient Federated Learning with Secure Aggregation and Differential Privacy

Wei-Ning Chen* Google Research

Christopher A. Choquette-Choo* Google Research

Peter Kairouz* Google Research

Abstract

Optimizing the privacy-utility-communication tradeoff is a key challenge for federated learning. Under distributed differential privacy (DP) via secure aggregation (SecAgg), we prove that the worst-case communication cost per client must be at least $\Omega\left(d \log\left(\frac{n^2\varepsilon^2}{d}\right)\right)$ to achieve $O\left(\frac{d}{n^2\varepsilon^2}\right)$ centralized error, which matches the error under central DP. Despite this bound, we leverage the near-sparse structure of model updates, evidenced through recent empirical studies, to obtain improved tradeoffs for distributed DP. In particular, we leverage linear compression methods, namely sketching, to attain compression rates of up to $50\times$ with no significant decrease in model test accuracy achieving a noise multiplier 0.5. Our work demonstrates that fundamental tradeoffs in differentially private federated learning can be drastically improved in practice.

1 Introduction

Federated learning (FL) is a widely used machine learning (ML) framework where multiple clients collaborate in learning a model under the coordination of a central server [12, 10]. One of the primary attractions of FL is that it provides data confidentiality and can provide a level of privacy to participating clients through data minimization: the raw client data never leaves the device, and only updates to models (e.g., gradient updates) are sent back to the central server. These model updates are more focused on the learning task at hand than is the raw data (i.e., they contain strictly no additional information about the client, and typically significantly less, compared to the raw data); further, the individual updates only need to be held ephemerally by the server. While these features can offer significant practical privacy improvements over centralizing all the training data, there is still no formal (worst-case) guarantee of privacy in this baseline federated learning model. Two key technologies for formalizing and strengthening FL's privacy guarantees are secure aggregation (SecAgg) [2], and distributed differential privacy (DP) [9, 5].

Focusing on a setting where SecAgg and distributed DP are used to ensure that the server only sees the aggregated and noised model updates, we prove a fundamental lower bound on the (worst-case) communication cost required to achieve optimal privacy-accuracy tradeoffs that match communication-unconstrained, central DP. Our lower bound essentially shows that for estimating the mean of n, d-dimensional vectors under ε distributed DP via SecAgg, $\Omega\left(d \log\left(\frac{n^2\varepsilon^2}{d}\right)\right)$ bits are necessary to achieve the optimal privacy-accuracy tradeoffs. This shows that, in the worst-case, there is no hope to reduce the communication cost of the distributed DP solution of [9]. We note that the recent works of [7, 6] show that an $O(\log(d))$ bits communication per client is sufficient under an ideal shuffled distributed DP model. Their compression methods require random sampling of parameters and are thus not compatible with SecAgg, which only allows for linear compression schemes.

*Equal contributions. Authors listed alphabetically.

Preprint. Under review.

To overcome the above-mentioned impossibility result, we leverage insights from recent works that indicate near-sparsity in the gradients [16, 13] and propose linear compression schemes that allow us to reduce the client-to-server communication cost without sacrificing utility. Experiments on EMNIST-62 demonstrate that for a CNN model with 1 million parameters, we can compress the client updates by $10\times$ without impacting its accuracy. When distributed DP is used, we can compress client updates by up to $50\times$. Yet, training a 0.2 million parameter model on EMNIST-62 fails to achieve similar accuracy. These results demonstrate that our scheme is not only compressing an overparameterized model and is instead beneficial for achieving performant (large) ML models. In particular, our results show that under practical considerations of FL we can achieve (much) better privacy-utility-communication tradeoffs than previously discovered by [9]. We discuss societal considerations in Appendix A.3.

1.1 Related Works

We consider federated learning with distributed DP via SecAgg, which consists of five major steps applied on the model update on the client side: (1) random rotation to spread out values and achieve a sub-Gaussian Distribution, (2) ℓ_2 clipping to bound the sensitivity (as inDP-SGD), (3) randomized rounding to move from continuous values (floats) to integers, (4) discrete Gaussian noise addition and modular clipping to ensure bounded precision, and (5) aggregation of noised client updates using SecAgg [2], returning the result to the server where these operations are undone in reverse order to obtain the aggregated model update. We explore how these tradeoffs are affected by linear compression operators (that commute with SecAgg), in particular, count sketches [4]. Most similar to our work is that of [15] and [8], which also explore sketching in FL. [15] use a count-median sketch which creates a biased gradient estimator. They prove convergence using error accumulation similar to [16] and demonstrate compression rates of up to 5x on CIFAR10 and CIFAR100 [11], and up to 3x on EMNIST [3]. [8] creates a new unbiased gradient sketch operator called HEAPRIX observing compression rates of nearly 12x on MNIST. Our work is crucially different from these because they do not consider DP or SecAgg—they only explore how linear compression affects the communication-utility tradeoff; we also use an unbiased count-mean sketch guaranteeing convergence.

2 Optimal Privacy-Utility-Communication Tradeoffs under SecAgg

2.1 Impossibility results for (worst-case) mean estimation

We first derive a communication lower bound for estimating the mean of n *arbitrary* vectors under SecAgg. Consider n clients each with data $x_i \in \mathbb{R}^d$ that satisfies $\|x_i\|_2 \leq 1$. A server estimates the mean $\bar{x} \triangleq \frac{1}{n}\sum_i x_i$ using a secure aggregation channel $\mathcal{W}_{\mathsf{SA}} : G^n \to G$, where G is a finite additive group that the SecAgg protocal operates on. To do so, each client applies a local encoding function $\mathcal{A}_i : \mathcal{X} \to G$, and the server collects

$$\mathcal{W}_{\mathsf{SA}}\left(\mathcal{A}_1(x_1), ..., \mathcal{A}_n(x_n)\right) = \sum_i \mathcal{A}_i(x_i),$$

where the summation is carried out under the finite (Abelian) group G. Usually we operate on a module M space, i.e. setting $G = (\mathbb{Z}_M)^m$ for some $m, M \in \mathbb{N}$. Finally, the server outputs an estimate $\hat{x}\left(\sum_i \mathcal{A}_i(x_i)\right)$ that minimizes the ℓ_2^2 error $\mathbb{E}\left[\|\hat{x} - \bar{x}\|_2^2\right]$, where the expectation is taken with respect to the (possible) randomness used by the estimator $\hat{x}$ and all local encoders $\{\mathcal{A}_i, i = 1, ..., n\}$. The following lemma bounds the ℓ_2^2 estimation error by the cardinality of G, showing that when $|G|$ is small, the resulting estimator must suffer from large error.

Lemma 2.1 (Lower Bound on the Estimation Error) *Given the above setup, the minimax estimation error is lower bounded by*

$$\inf_{(\mathcal{A}^n, \hat{x})} \sup_{\bar{x}:\|\bar{x}\|_2 \leq 1} \mathbb{E}\left[\|\hat{x} - \bar{x}\|_2^2\right] \geq \left(\frac{1}{|G|}\right)^{2/d} \tag{1}$$

We defer the proof to Appendix A.1. Note that to transmit the encoded message $\mathcal{A}_i(x_i) \in G$, client i needs to pay $\log(|G|)$ bits of communication. Therefore Lemma 2.1 implies that, to achieve

the centralized error $O\left(\frac{d}{n^2\varepsilon^2}\right)$ (assuming $\frac{d}{n^2\varepsilon^2} \leq 1$), the per-client communication complexity is $\Omega\left(d \log\left(\frac{n^2\varepsilon^2}{d}\right)\right) = \tilde{\Omega}\left(d\right)$. This communication lower bound is achieved by the distributed discrete Gaussian mechanism (up to some logarithmic constants, see [9, Theorem 2]). This result indicates that there is minimal room for improvement on the communication cost—each client must communicate approximately the size of the model on each update.

2.2 Reducing Communication Costs by Leveraging Near-Sparse Structures

Notice that the results of Lemma 2.1 are based on the cardinality of $|G|$, the (securely) aggregated estimate of the model update. This means that when the server update estimate is sparse (or near-sparse) such that $G' = G$ (or $G' \approx G$) satisfies $|G'| \ll d$ we can still compress client transmission. Prior work in sparse SGD [16, 13] shows that model updates in centralized settings are near-sparse in that high levels of compression can be attained. In the FL setting, this translates to each aggregand's (each client's model update, or x_i above) being near-sparse. Our thesis is that this extends to the aggregate update $\bar{x}$. We emphasize that this is a non-trivial extension. For example, consider the case where each aggregand x_i is exact k-sparse such that $\|x_i\|_0 = k$. In this case, the aggregate can be as dense as $\min(n \cdot k, d) = O(d)$.

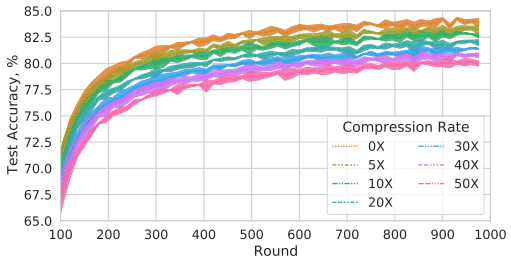
Under the assumption that $\bar{x}$ is still near-sparse, we propose the following protocol. We follow the standard Federated Averaging protocol, starting with random initial weights W_0. For R rounds, we subsample n clients each round and broadcast the current model weights W_r. We train for E epochs locally on each client and calculate the i-th client model update as $\Delta(W_r - W_r^i)$. To transmit and aggregate this update back to the server we propose two key steps: (1) concatenating of the model weights to achieve a single gradient vector and (2) compressing the concatenated weights to a latent dimension of shape $[sketch\_length,\ sketch\_width]$ using a count-sketch (with the coordinate index as the unique value and the coordinate value as the frequency). Following, we proceed with the distributed DP protocol from [9]. On the server, after SecAgg and the steps outlined by [9], we compute a final decompression operation using a count-mean sketch decoding (see Appendix A.2 Algorithm 1) and undo the weight concatenation obtaining the structured model update. Thus, our protocol obtains a compression rate factor that reduces each client's communication cost by $d/(sketch_length \cdot sketch_width)$x.

3 Empirical Analysis

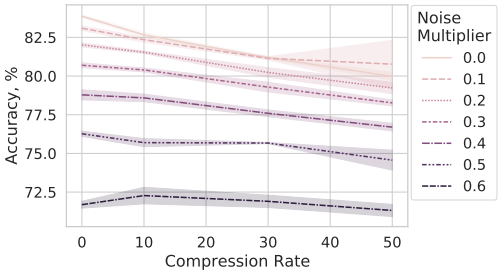
We run experiments on the full Federated EMNIST dataset [3], a common benchmark for FL tasks. The dataset has 62 classes, 3400 clients, with each user holding both a train and test set of examples. In total, there are $671,585$ training examples and $77,483$ test examples. Inputs are single-channel $(28, 28)$ images. We sample $n = 100$ clients per round for a total $R = 1000$ rounds. We pick $E = 1$ local training epochs per client and use a local batch size of 20. The server uses Stochastic Gradient Descent with learning rate 1 and momentum of 0.9 [14]; the client uses a learning rate of 0.01 without momentum. For distributed DP, we use the geometric adaptive clipping of [1] with an initial ℓ_2 clipping norm 0.1 and a target quantile of 0.8. We use the same procedure as [9] and flatten using the Discrete Fourier Transform, pick $\beta = \exp -0.5$ as the conditional randomized rounding bias, and use a modular clipping target probability of $6.33\text{e}{-5}$ or ≈ 4 standard deviations at the server (assuming normally distributed updates). We experiment with two models: 'large' is an ≈ 1 million parameter Convolutional Neural Network (CNN) used by [9]. This model has two convolutional layers followed by two dense layers (see Appendix A.5 Figure 4). 'Small' is an $\approx 200,000$ parameter model with 3 dense layers (see Appendix A.5 Figure 5). We repeat all experiments with 3 different random seeds.

3.1 Analyzing the Privacy-Utility-Communication Tradeoff

As a baseline, we first explore the communication-accuracy tradeoffs of our count-sketch method without DP. Observing Figure 1a, we see that the 'large' model under all compression rates converges to a well-trained model. However, as we increase the compression rate, we observe a constant decrease in the model's learning as shown by the consistent decrease in test-time error throughout training. The top line of Figure 1b shows the model performance during training relative to the baseline without compression. Observing this plot, we see that large compression rates can lead to a significant decrease in test accuracy of up to ≈ 2.5 percentage points. In particular, we observe



(a) **Test Accuracy during training for various compression rates and No DP.** Observe that compression rates $> 10\times$ lead to a > 1 %-point decrease in the converged test accuracy.



(b) **Converged model test accuracy with varied noise multipliers.** Observe that high noise multipliers can achieve higher compression without significant degradation in test accuracy.

Figure 1: **Analyzing the privacy-utility-communication tradeoff.**

that a maximum compression rate of 10x is possible without significantly degrading the converged model test accuracy (i.e., beyond 1 percentage point). In these experiments, we varied both the $sketch\_length$ and the $sketch\_width$ but found that the $sketch\_length$ had little impact on thet test accuracy so long as it was sufficiently large. Thus, we fixed it to 20 for all experiments and vary only the $sketch\_width$.

To analyze the privacy-utility-communication tradeoff, we vary the both the compression rate and the noise multiplier; the latter can be directly converted to a user-level (ε, δ)-DP guarantee. Observing Figure 1b, increasing the noise multiplier decreases the converged model test accuracy, as expected. Interestingly, we see that as the noise multiplier increases, there is a decreased impact of compression on the converged model test accuracy. However, an ML practitioner will generally aim to maximize test performance under a *pre-specified noise multiplier*. In this case, it is more interesting to look at how compression impacts learning for a given noise multiplier. When we analyze Figure 1b in this way, we observe an interesting benefit to linear compression with distributed DP: for a larger noise multiplier, the compression rate can be pushed significantly higher with no significant decrease in test accuracy, up to 50x for a noise multiplier of 0.6.

3.2 Impact of Model Size on the Privacy-Utility-Communication tradeoff

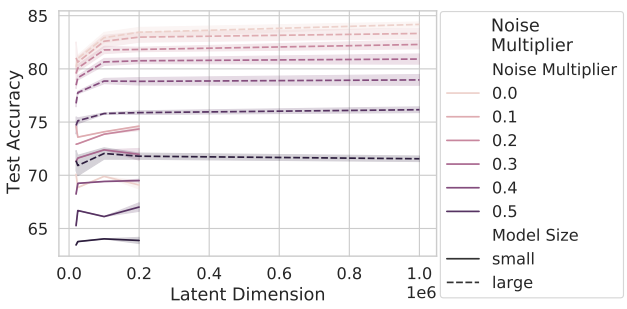


Figure 2: **Impact of the number of parameters on the privacy-utility-communication tradeoff.**

In Section 3.1 above, we observed that 'large' models trained to satisfy an ε-DP guarantee can get compression *for free*—there is no significant decrease in converged test accuracy. However, our experiments beg to question: "If we can compress a model during learning, could we have trained a smaller model?" Our results in Figure 2 demonstrate that we cannot (without sacrificing performance). In particular, we observe that the 'large' model with high levels of compression outperforms the 'small' model to a compression of the same sized latent dimension. For example, we see that the 'large' model with a noise multiplier of 0.4 and a $50\times$ compression factor matches the performance of the 'small' model. This result indicates that our protocol is in fact beneficial to learning: for a fixed communication bandwidth, we can train a larger more performant model with the same privacy guarantees.

References

[1] Galen Andrew, Om Thakkar, H Brendan McMahan, and Swaroop Ramaswamy. Differentially private learning with adaptive clipping. *arXiv preprint arXiv:1905.03871*, 2019.

[2] K. A. 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.

[3] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

[4] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *International Colloquium on Automata, Languages, and Programming*, pages 693–703. Springer, 2002.

[5] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 486–503. Springer, 2006.

[6] Antonious Girgis, Deepesh Data, Suhas Diggavi, Peter Kairouz, and Ananda Theertha Suresh. Shuffled model of differential privacy in federated learning. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 2521–2529. PMLR, 13–15 Apr 2021.

[7] Antonious M Girgis, Deepesh Data, Suhas Diggavi, Peter Kairouz, and Ananda Theertha Suresh. Shuffled model of federated learning: Privacy, communication and accuracy trade-offs. *arXiv preprint arXiv:2008.07180*, 2020.

[8] Farzin Haddadpour, Belhal Karimi, Ping Li, and Xiaoyun Li. Fedsketch: Communication-efficient and private federated learning via sketching. *arXiv preprint arXiv:2008.04975*, 2020.

[9] Peter Kairouz, Ziyu Liu, and Thomas Steinke. The distributed discrete gaussian mechanism for federated learning with secure aggregation. *arXiv preprint arXiv:2102.06387*, 2021.

[10] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konecný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[11] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[12] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[13] Leighton Pate Barnes, Huseyin A Inan, Berivan Isik, and Ayfer Ozgur. rtop-k: A statistical estimation approach to distributed sgd. *arXiv e-prints*, pages arXiv–2005, 2020.

[14] Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17, 1964.

[15] Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph Gonzalez, and Raman Arora. Fetchsgd: Communication-efficient federated learning with sketching. In *International Conference on Machine Learning*, pages 8253–8265. PMLR, 2020.

[16] Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified sgd with memory. *arXiv preprint arXiv:1809.07599*, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] All data is already open source and all instructions are included in the main-text. All code will be uploaded if accepted.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? **[TODO]**
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

A.1 Proof for Lemma 2.1

Proof A.1 *First, we claim that if there exists a scheme* $(\mathcal{A}^n, \hat{x})$ *such that*

$$\mathbb{E}\left[\|\hat{x} - \bar{x}\|_2^2\right] \leq r^2, \text{ for some } r > 0, \tag{2}$$

then there exists a r-covering $\mathcal{C}(r)$ over the ℓ_2 unit ball $\mathcal{B}_d(1) \triangleq \left\{x \in \mathbb{R}^d : \|x\|_2 \leq 1\right\}$ so that $|\mathcal{C}(r)| \leq |G|$.

To see this, observe that if $(\mathcal{A}^n, \hat{x})$ satisfies (2), then $\{\mathbb{E}\left[\hat{x}(g)\right], g \in G\}$ forms a r-covering of $\mathcal{B}_d(1)$. Since for any $\bar{x} = \sum_i x_i \in \mathcal{B}_d(1)$, by setting $g = \sum_i \mathcal{A}_i(x_i)$ we obtain

$$\|\mathbb{E}\left[\hat{x}\right] - \bar{x}\|_2^2 \overset{(a)}{\leq} \mathbb{E}\left[\|\hat{x} - \bar{x}\|_2^2\right] \leq r^2,$$

where (a) holds by Jensen's inequality.

On the other hand, for any r-covering, it holds that $|\mathcal{C}(r)| \geq \frac{\mathsf{vol}(\mathcal{B}_d(1))}{\mathsf{vol}(\mathcal{B}_d(r))} = \left(\frac{1}{r}\right)^d$. Thus, as long as $|G| \leq \left(\frac{1}{r}\right)^d$, we must have $\mathbb{E}\left[\|\hat{x} - \bar{x}\|_2^2\right] \geq r^2$, or equivalently

$$\mathbb{E}\left[\|\hat{x} - \bar{x}\|_2^2\right] \geq \left(\frac{1}{|G|}\right)^{2/d}. \tag{3}$$

A.2 Additional Algorithms

Algorithm 1 Gradient Count Sketch Decoding

Require: Sketch S, gradient vector size d, shared seed $seed$

```
gradient_estimate ← zeros(d)
for all hash_index in [0, ··· , S.length], in parallel do
    hash_seed ← hash_index + seed
    indices ← random_uniform(0, S.width, hash_seed)
    signs ← random_choice([−1, 1], hash_seed)
    gradient_estimate+ = signs ∗ S[hash_index, indices]
```

A.3 Societal Considerations

Our work explores private learning from distributed data. In general, this enables tasks like (federated) machine learning without compromising the privacy of users who contribute data to these protocols. However, privacy guarantees are complex and the relationship between optimized privacy metrics like ε-DP with the practical privacy leakage are not well understood. Because of this, works that leverage private learning protocols but do not guarantee a tight (ε, δ)-DP bound may provide a false sense of privacy for user data.

A.4 Additional Figures

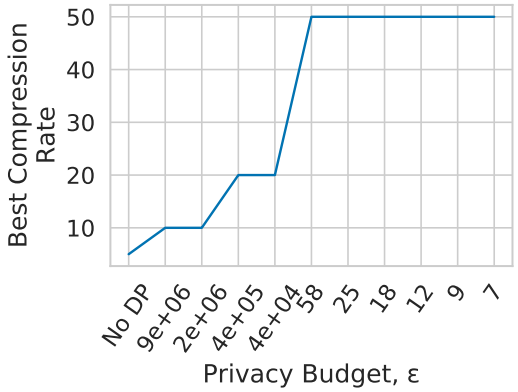


Figure 3: **Best compression rate for each** ε with a maximum 1 percentage-point drop in converged test accuracy.

A.5 Model Architectures

```
Model: "Large Model: 1M parameter CNN"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_1 (MaxPooling  (None, 13, 13, 32)        0
2D)
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
dropout_2 (Dropout)          (None, 11, 11, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 7744)              0
_________________________________________________________________
dense_8 (Dense)              (None, 128)               991360
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_9 (Dense)              (None, 62)                7998
=================================================================
Total params: 1,018,174
Trainable params: 1,018,174
Non-trainable params: 0
_________________________________________________________________
```

Figure 4: 'Large' model architecture.

```
Model: "Small Model, 0.2M parameter Dense Network"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
reshape_2 (Reshape)          (None, 784)               0
_________________________________________________________________
dense_10 (Dense)             (None, 200)               157000
_________________________________________________________________
dense_11 (Dense)             (None, 200)               40200
_________________________________________________________________
dense_12 (Dense)             (None, 62)                12462
=================================================================
Total params: 209,662
Trainable params: 209,662
Non-trainable params: 0
_________________________________________________________________
```

Figure 5: 'Small' model architecture.