TOFU: Toward Obfuscated Federated Updates by Encoding Weight Updates Into Gradients From Proxy Data

MANISH NAGARAJ, (Graduate Student Member, IEEE), ISHA GARG, AND KAUSHIK ROY, (Fellow, IEEE)
School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, USA

Corresponding author: Manish Nagaraj (mnagara@purdue.edu)

This work was supported in part by the Center for the Co-Design of Cognitive Systems (CoCoSYS), one of the centers in the Joint University Microelectronics Program (JUMP) 2.0, a Semiconductor Research Corporation (SRC) program sponsored by the Defense Advanced Research Projects Agency (DARPA), by the Semiconductor Research Corporation, the National Science Foundation, Intel Corporation, the Department of Defense (DoD) Vannevar Bush Fellowship, and by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement W911NF-16-3-0001.

ABSTRACT
Advances in Federated Learning and an abundance of user data have enabled rich collaborative learning between multiple clients, without sharing user data. This is done via a central server that aggregates learning in the form of weight updates. However, this comes at the cost of repeated expensive communication between the clients and the server, and concerns about compromised user privacy. The inversion of gradients into the data that generated them is termed data leakage. Encryption techniques can be used to counter this leakage but at added expense. To address these challenges of communication efficiency and privacy, we propose TOFU, a novel algorithm that generates proxy data that encodes the weight updates for each client in its gradients. Instead of weight updates, this proxy data is now shared. Since input data is far lower in dimensional complexity than weights, this encoding allows us to send much lesser data per communication round. Additionally, the proxy data resembles noise and even perfect reconstruction from data leakage attacks would invert the decoded gradients into unrecognizable noise, enhancing privacy. We show that TOFU enables learning with less than 1% and 7% accuracy drops on MNIST and CIFAR-10 datasets, respectively. This drop can be recovered via a few rounds of expensive encrypted gradient exchange. This enables us to learn to near-full accuracy in a federated setup, while being 4× and 6× more communication efficient than the standard Federated Averaging algorithm on MNIST and CIFAR-10, respectively.

INDEX TERMS
Communication efficiency, federated learning, gradient matching, privacy-preserved learning, synthetic data.

I. INTRODUCTION
Federated learning is the regime in which many devices have access to localized data and communicate with each other either directly or through a central node. The goal is to improve their learning abilities collaboratively, without sharing data. Here, we focus on the centralized setting, in which each device or ‘client’ learns on the data available to it and communicates the weight updates to a central node or ‘server’, which aggregates the updates it receives from all the clients. The server propagates the aggregated update back to each client, thus enabling collaborative learning from data available to all devices, without actually sharing the data. The abundance of user data has enabled rich complex learning. However, this comes at the cost of increased computational or communication costs between the clients and the server, and with increasing concerns about compromised user privacy. Privacy of user data is a growing concern, and standard federated averaging techniques are vulnerable to data leakage by inverting gradients into the data that generated them [1], [2], [3], [4], [5]. Gradients can be encrypted to preserve privacy, but incur further communication overhead [6].
In this work, we focus on the communication between the clients and the server, a critical point for both communication and data leakage. Traditionally, in every communication round, each client shares its weight updates with the servers. To enable complex learning, the models are getting larger, growing to many millions of parameters [7], [8]. To put things in context, a VGG13 model has 9.4 million parameters, resulting in 36 MB of data being shared per communication round, per device. Since each device only has limited data, the number of rounds needed for the server to reach convergence is orders of magnitude more than those needed by the individual clients, further heightening the communication cost and opportunities for privacy leaks. This cost quickly grows prohibitive in resource-constrained settings with limited bandwidth.

To address these concerns, we propose TOFU, a novel algorithm that works Towards Obfuscated Federated Updates, outlined pictorially in Figure 1. Here, each client generates synthetic proxy data whose combined gradient captures the weight update and communicates this data instead of the weights. This mitigates two issues simultaneously. Data is much lower dimensional than gradients. For context, CIFAR-10 images only have 3072 pixels, and we show that TOFU needs under 100 images to capture the weight updates well. Sending these images instead of the weight updates for VGG13 results in an order of magnitude reduction in communicated costs per round. Additionally, the synthetic data resembles noise, and existing techniques would invert the gradients to this noise rather than the true data, thus enhancing privacy. To further improve communication efficiency and encourage the synthetic images to differ from the true data distribution, we show that our method can approximate the gradient well even with images that are downsampled by 4 ×, or reduced to a single channel. The synthetic images are visualized in Figure 2.

Since our method approximates gradients to reduce communication costs and enhance privacy, it results in a slight accuracy drop. We exchange proxy data for most of the communication rounds, which are tolerant to noisy updates. Closer to convergence, updates are more precise and approximations are harmful. In these few communication rounds, we recover any accuracy drop by sharing the true full weight updates. In this phase, care needs to be taken to ensure privacy via expensive encryption techniques. Since this sensitive phase consists of far fewer communication rounds than the non-sensitive learning phase, the overhead resulting from this is countered by the communication efficiency achieved by sharing synthetic data instead of weight updates for most of the communication rounds. We show that we need only 3 and 15 full weight update rounds for MNIST and CIFAR-10, respectively, to recover any drop in accuracy.

This proposed hybrid approach provides both communication efficiency and privacy, without any loss in accuracy. We demonstrate TOFU on the CIFAR-10 dataset in single device setups and show that we can learn with 3% accuracy drop while communicating 17 × lesser parameters on average. We extend this to a federated setup, with data distributed in an IID (Independent and Identically Distributed) fashion. We show that with a few additional rounds of full weight update, we can learn to accuracies comparable to FedAvg while achieving up to 4.6 × and 6.8 × better communication efficiency on MNIST and CIFAR-10, respectively. We emphasize that TOFU will result in increasing gains with the increasing complexity of the models since the size of weight updates will grow but the image size stays constant.

The rest of this manuscript is organized as follows. In Section II, we provide a brief overview of existing works on Federated Learning, specifically works that focus on improving the efficiency, privacy preservation, and accuracy of federated platforms. Section III outlines the desired properties of the synthetic dataset, the algorithm to create
it, and the tradeoff between communication efficiency, privacy, and accuracy. Section IV describes the experiments performed and provides an analysis of the results obtained. We conclude this manuscript in Section V.

II. BACKGROUND

Reference [10] pioneered the field of Federated Learning and proposed the first algorithm for distributed private training, called FedAvg, which serves as our baseline. Here, only weight updates are shared with the server, which aggregates updates from all clients and shares them back with each client. There are two research thrusts, that depend on whether the local data present at any client is distributed in an IID fashion or not. We focus on the IID setting in this work and direct readers to [11] for a better survey on non-IID methods. In this section, we focus on and discuss the relevant works on three key aspects in the IID setting: efficiency, privacy, and accuracy, and also provide an overview of the works that use distillation techniques to improve communication efficiency.

A. EFFICIENCY

Federated learning has two key areas of inefficiency: communication cost, both from client to server (up-communication) and from server to client (down-communication), and computational cost. The most potential for impact comes with decreasing client to server communication [11]. In our work, we target both up- and down-communication efficiency. Related works include quantization or sparsification of the weight updates [12], [13], [14], [15]. While they significantly improve communication efficiency, there have been concerns raised [11] about their compatibility with secure aggregation [6] and differential privacy techniques [16]. Our method can be thought of as an indirect compression, by encoding updates into proxy inputs. Our proxy images are amenable to encryption, and can potentially be further quantized, resulting in additional savings. In this work, we focus on showing that it is possible to encode gradients into synthetic fake-looking data and still enable learning. Other methods restrict the structure of updates, such as to a low rank or a sparse matrix [12], or split the final network between the client and the server [17]. We impose no constraints on learning, and focus on the standard case where each client has a synchronized model and equal accuracy on queries from any other client’s dataset.

B. PRIVACY

Recent methods such as Inverting Gradients (IG) [2] and Deep Leakage from Gradients (DLG) [1] have shown that gradients can be inverted into the training images that generated them, violating user data privacy. Gradient inversion in alternative spaces (GIAS) [18] utilizes a generative model pretrained on the data distribution to take the target gradient as input and outputs reconstructed images. GradInversion [3] improved upon IG by introducing better fidelity metrics in the objective to recover images from federated scenarios with larger batch sizes, and more complex models and datasets such as ResNets and ImageNet. This is cause for concern, and we circumvent it by showing that our proxy data looks like noise, and hence even perfect inversion by these techniques would only resemble noise. Reference [4] show that altering the model architectures minimally allows the server to obtain user data without solving complex optimization problems. They also show that modifying only the larger linear layers can help recover user data. Methods to secure gradients from attack involve encryption and differential privacy techniques that add additional computational expense [6], [16]. These methods are compatible with our proxy data, should the need for extra encryption arise. Additionally, encrypting our proxy data will be less costly since standard encryption costs are proportional to the size of the vector being encrypted [6].

C. ACCURACY

Efforts to increase accuracy often focus on variance-reduced Stochastic Gradient Descent (SGD) [19], [20] or on adaptive optimization and aggregation techniques [21], [22]. Astraea [23] reschedules client participation based on the KL divergence of their data distribution in order to overcome data distribution imbalances to improve accuracy in federated settings. Our method is orthogonal and compatible with such techniques.

D. DISTILLATION

Recently, there has been interest in one-shot federated learning, wherein there is only one communication round. An approach that is similar to ours, called DOSFL [24] focuses on this setting. It is based on the dataset distillation technique, in which the entire dataset is distilled into synthetic data. DOSFL uses this to distill the local data of each client and share that for one-shot learning. There are a few key differences between our method for synthetic data generation and dataset distillation. We generate proxy data that aligns its gradients to a desired weight update, whereas dataset distillation optimizes data for accuracy after learning on it. Dataset Distillation shows very large drops in accuracy for CIFAR-10 dataset ($\sim 26\%$) versus our single device results (Section IV-A), which shows an average of 3% drop. DOSFL gets impressive results on MNIST, especially for a single round of communication but does not show results on larger datasets like CIFAR-10, presumably due to the significant drop in the baseline technique of dataset distillation. In parallel with the development of this work, [26] came up with an extension of dataset distillation that precomputes training trajectories from an expert and saves them to guide the synthetic image creation. While the work overlaps with ours in concept, they focus on creating coresets to make training efficient, and we focus on improving communication efficiency in federated learning.

III. METHODOLOGY

In this section, we first describe the desired properties of the synthetic dataset in Section III-A. We then outline the algorithm to create the synthetic dataset in Sections III-B
and III-C, and finally discuss the tradeoff between communication efficiency, privacy, and accuracy in Section III-D.

A. DESIDERATA OF THE SYNTHETIC DATASET

The generated synthetic dataset should have two properties - (a) it must be small in size in order to ensure communication efficiency and (b) it should not resemble the true data to ensure that data leakage attacks are unable to invert the proxy gradients into real data, thus ensuring privacy of the real data. We discuss these in more detail next.

1) COMMUNICATION EFFICIENCY

Input data is much lower in dimensional complexity than gradients (for instance, 3072 parameters per image in CIFAR-10 compared to 9.4 million parameters for sending VGG13 weight updates). This allows us to attain the first goal of efficient communication. We experimentally show that 64 images give us good results, which allows us to send ~ 50× lesser data per communication round as compared to the weight updates of VGG13. For improving efficiency, we also experiment with images that have (a) the 3 channels reduced to a single channel, replicated across all 3 channels, and (b) the height and width reduced by 2×, which is then upsampled with their nearest neighbor values as part of pre-processing. This adds a savings of 3× and 4×, respectively.

2) ENHANCED PRIVACY FROM DATA LEAKAGE

To attain the goal of privacy, we rely on the high dimensional, non-linear nature of neural networks to generate images that resemble noise to the human eye. We distill the weight update of a client after learning on many minibatches into a single minibatch of synthetic images. Combining weighted gradients is not the same as combining inputs, and we observe that condensing the learning from many images into a smaller set results in images that visually do not conform to the true data distribution. The resulting weighted gradient we generate is an approximation of the true weight update. Figure 3 shows that while full gradient exchanges are vulnerable to the various attacks ([1], [2], [3], [118]) discussed in Section II, the lossy compression from our method buffers our synthetic gradients from data leakage. If these attacks were to invert the images perfectly, the inverted images would still look like noise, circumventing data leakage. We further corroborate this, by pointing out that all the attacks (except DLG) converge to very similar-looking noisy data. This ensures that we are not just using bad hyperparameters for attacks, but rather there is enhanced privacy due to our formulation.

We employ some additional tricks to encourage the obscurity of generated images. Many attacks against privacy assume the availability of one-hot labels, or reconstruct one label per image. We use soft labels to further discourage reconstruction. Additionally, we weigh the gradients differently into the combined final gradient so that no true gradient is well represented. We distill the updates from a large number of minibatches (to the tune of a whole epoch) down to just a handful of images. While this incurs an accuracy loss, it results in images that do not resemble the real data distribution at all. We can further force a downsampling prior to the images, in either the number of channels or the image height and width, which allows the image distribution to differ from the original one even more. We provide visualizations of the synthetic images generated by TOFU at various stages of the training process in Appendix B.

3) TRADEOFFS

The tradeoff cost associated is two-fold: a) the clients have added computational complexity to create the synthetic data and b) the communication rounds needed for convergence increase since we introduce some error in the weight updates. We emphasize that our method is better applied to use-cases where the clients have computational resources but are limited in communication bandwidth or cost. Furthermore, communication efficiency has been identified as the major efficiency bottleneck, with the potential for most impact [11]. We account for the latter tradeoff of increased communication rounds when reporting our final efficiency ratios and provide an analysis of the communication and computational overheads of TOFU in Section IV-D.

B. CREATING THE SYNTHETIC DATASET

We now detail the algorithm that distills the change in model parameters into a synthetic dataset during training. Taking inspiration from the IG attack, we optimize synthetic data to align the resulting gradient direction with the true weight update. To formalize, let this true weight update attained after
a client learns on its true data be referred to as $\mathbf{U}_{\text{real}}$. We want to generate a synthetic dataset,

$$D_{\text{syn}} = \{(x_{\text{syn}}, y_{\text{syn}}, \alpha_{\text{syn}}); \ i = 1 \ldots N\}$$

where $N$ is the number of images in the synthetic dataset. $x_{\text{syn}}$ and $y_{\text{syn}}$ refer to the $i^{th}$ image and soft label respectively. The goal of reconstruction is that the combined gradient obtained upon forward and back-propagating all $\{x_{\text{syn}}, y_{\text{syn}}\}$ is aligned to the true weight update, $\mathbf{U}_{\text{real}}$. Each synthetic datapoint generates a single gradient direction, and with $N$ datapoints in our synthetic dataset, we generate $N$ different gradient directions. Traditionally, if we were to treat these $N$ images as a mini-batch, we would average the $N$ gradients. However, we take a weighted average of the gradients, allowing us to span a larger space. We jointly optimize these weights, referred to as $\alpha_{\text{syn}}$; $\sum \alpha_{\text{syn}} = 1$, along with the images and soft label.

Next, we show that weighing the gradients of each image in the backward pass is the same as weighing the losses from each image on the forward pass, since derivative and summation can be interchanged. Formally, let $\theta$ be the weights of the model, and $\theta(x)$ the output of the model. Let the loss per synthetic datapoint, denoted by $L(\theta(x_{\text{syn}}), y_{\text{syn}})$, be weighted by the respective $\alpha_{\text{syn}}$, and summed into $L_{\text{syn}}$, the overall loss of the synthetic dataset. Let the resulting gradient from backpropagating $L_{\text{syn}}$ be $\mathbf{U}_{\text{syn}}$. Backpropagating $L_{\text{syn}}$ results in the desired weighted average of the individual gradients per datapoint, as shown:

$$L_{\text{syn}} = \sum_i \alpha_i L(\theta(x_{\text{syn}}), y_{\text{syn}})$$

$$\mathbf{U}_{\text{syn}} = \frac{\nabla L_{\text{syn}}}{\nabla \theta} = \sum_i \alpha_i \frac{\nabla L(\theta(x_{\text{syn}}), y_{\text{syn}})}{\nabla \theta}$$

Standard cross entropy loss is used to calculate gradients from both the true and the synthetic data. The synthetic data is optimized by minimizing the reconstruction loss, $R_{\text{loss}}$, which is the cosine similarity between the true update $\mathbf{U}_{\text{real}}$, and the synthetic update $\mathbf{U}_{\text{syn}}$.

$$R_{\text{loss}} = \left(1 - \frac{< \mathbf{U}_{\text{real}} \cdot \mathbf{U}_{\text{syn}} >}{||\mathbf{U}_{\text{real}}|| \cdot ||\mathbf{U}_{\text{syn}}||}\right)$$

Since minimizing $R_{\text{loss}}$ only aligns the directions of gradients, we additionally send scaling values for each layer from $\mathbf{U}_{\text{real}}$ to scale up $\mathbf{U}_{\text{syn}}$. To avoid cluttering notation, we leave this out since it only adds extra parameters equal to the number of layers. In addition, we also use soft labels instead of the hard labels used in the real dataset. This provides more flexibility for the optimization algorithm to create a better alignment between synthetic and true updates, and discourages attacks like IG, which rely on one-hot labels. We use Adam [27] to optimize the randomly initialized images to generate a gradient that aligns with the true weight updates. We use learning rates 0.1 for images, labels and $\alpha$s, for 1000 iterations, decayed by a factor of 0.1 at the $375^{th}$, $625^{th}$, and the $875^{th}$ iteration. For the downsampling experiments, since the network expects inputs of size $32 \times 32 \times 3$, we replicate the single channel along all dimension in the grayscale case, and perform nearest neighbor upsampling before feeding in the images for the case with reduced image size.

C. TOFU: THE FEDERATED LEARNING ALGORITHM

We now put all the parts together and describe how we utilize the synthesized dataset to enable communication efficient and private federated learning. All clients and the server have the same model initialization before the learning phase starts. Every client first trains on its private local data for a few minibatches and determines the true weight update, $\mathbf{U}_{\text{real}}$, as the difference between the starting and ending weights. This true weight update is encoded using synthetic data ($D_{\text{syn}}$) as described in Section III-B. $D_{\text{syn}}$ and then communicated in lieu of weight updates to the server. The server decodes the information by performing a single forward and backward pass to get the encoded weight update. The server repeats this for proxy data received from all clients and averages the decoded updates. To ensure efficiency during down-communication as well, the server encodes its own weight update due to aggregation into proxy data, and sends this back to all clients. The clients then update their local models by decoding this information. The process is repeated until convergence. This is summarized in Algorithm 3 in the Appendix A.

D. EFFICIENCY-PRIVACY-ACCURACY TRADEOFF

The weight update statistics change with accumulation over different number of batches and convergence progress. We now introduce the various hyperparameters that need to be tuned, and their effect on accuracy, privacy and efficiency.

1) NUMBER OF SYNTHETIC DATAPONTS (NIMGS)

The size of the synthetic dataset transmitted per communication round has a direct impact on privacy, communication efficiency, and accuracy. While a larger synthetic dataset provides better accuracy as the encoding will be closer to the true weight update, the communication efficiency of the algorithm decreases since we have to communicate more data. We note that using 64-128 datapoints gives us the best empirical results. We see larger approximation errors with smaller number of datapoints, buffering us more against attacks.

2) SYNTHESIS FREQUENCY (SYNFREQ)

This denotes how many minibatches of weight updates should be accumulated by the client before communicating with the server. In FedAvg, this is usually one epoch. A very large Synfreq results in larger accuracy drops, since a large accumulated weight cannot be well represented by few synthetic images, but allows for enhanced privacy due to larger approximations. However, it degrades efficiency since we have to communicate more often per epoch.
3) PHASES TO IMPROVE ACCURACY (SWITCH)

In the initial phase of learning, the gradient step is quite error tolerant since there is a strong direction of descent. For the single device experiments, after a few communication rounds for warm-up, we empirically see better results by scaling the learning rate by the reconstruction error. This is implemented by scaling $V_{\text{syn}}$ by $(1-R_{\text{loss}})$, capturing the cosine similarity between the true and the synthetic update. This enables small steps to be taken if the synthetic data could not approximate the true update well. For single-device experiments, we switch from warm-up to this scaled learning rate phase at 200 communication rounds, and referred to it as switch 1.

In the Federated setting, we did not notice much improvement from this switch empirically, and thus leave it out for simplicity. However, we note that two sets of encoding are required now, for both up-communication and down-communication, and hence, we see more accuracy drop than the single device case. To counter this, we end with a few communication rounds of full weight update exchange to regain any accuracy loss. To ensure privacy, we recommend expensive encryption of the weight updates here. Since it consists of very few rounds (under 15), we do not sacrifice efficiency. We call the communication round where we switch to this final phase as switch 2 and mark it by a star in the learning curves shown in the Appendix C, and mention them in the corresponding hyperparameter sections. The efficiency savings we report take into account these rounds of expensive full gradient exchange as well. To summarize, for single device experiments, we have a brief warm-up phase of a few hundred communication rounds, and then switch to scaling learning rate by reconstruction error. For the federated setup, we do not scale the learning rate by communication round, and instead have a brief full weight update exchange phase of up to 15 communication rounds at the end of training.

4) EXPERIMENTING WITH IMAGE SIZES

We want to encourage synthetic data to resemble the true data distribution even lesser, while relying on our optimization algorithm to tune them to get a good match between the target gradient and the synthetic gradient. To do this, and to further enhance efficiency, we experiment with enforcing two downsampling priors on the synthetic images. The CIFAR-10 images are of size $32 \times 32 \times 3$, and this is also the size of our synthetic data. In one experiment, we downsample the number of channels in synthetic images from 3 to 1. The single channel is replicated across the 3 dimensions before being fed into the network for synthetic gradient calculation. In the other experiment, the synthetic images are sized $16 \times 16 \times 3$ with the height and width upsamlesd to the correct size by nearest neighbor upsampling before being fed into the network. We show a comparison of what the true images, the synthetic full size images, the grayscale images, and the downsampled images look like in Figure 2. In Section IV-A, we show that we can successfully learn with all three kinds of synthetic images with an average accuracy drop of 3% for full sized synthetic images, 5% for single-channeled images and 3.5% for images with width and length halved.

### IV. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, we first demonstrate TOFU on a single device setup to show that privacy-preserved learning with only synthetic data is possible in Section IV-A. This setup can be thought of as a federated setup with only 1 client and no down-communication. We initialize two copies of the same network with the same weights. Network 1 represents the client, and learns on real data for $Synfreq$ number of batches, generating $Nimgs$ number of synthetic datapoints to send to Network 2, which emulates the server. Network 2 only learns on the synthetic data. Post communication, both networks have the same weights since Network 1 knows how the synthetic data is going to update Network 2 and resets its own weights accordingly. For the single device experiments, we focus on getting the maximum accuracy from purely synthetic data, and hence we do not employ the final phase of full weight update exchanges. We first show the results of learning with similarly sized images as the real data, and then introduce priors that recreate the gradient via single channel images and images with width and height downsampled by 2 each.

We then extend it to multiple clients in a federated setup in Section IV-B. This has two encoding phases, the first carried out by each client to transmit their updates to the server (up-communication), and the second carried out by the server after aggregation from all clients (down-communication). Down-communication ensures that the weights of all clients and the server remain in sync after the end of each communication round. The experiments for the Federated setup include a few rounds of full gradient exchange at the end in order to circumvent any accuracy drop, and the emphasis in these experiments is on efficiency. We then

| Synthetic | Original | Grayscale | Downsampled |
|-----------|----------|-----------|-------------|
| Image Size | 32x32x3 | 32x32x1 | 16x16x3 |
| Max. Acc. (Comm.) | 81.24 | 82.39 | 85.74 |
| Max. Comm. Eff. | 97x | 45x | 30x |
| Max. Acc. Comm. Eff. | 82.61 | 84.14 | 55 |
| Varying $Nimgs$ @ Synfreq = 200 |
| 32 | 84.05 | 81.24 | 82.61 |
| 64 | 85.78 | 82.39 | 84.14 |
| 96 | 86.05 | 83.63 | 85.29 |
| 128 | 86.81 | 84.09 | 85.74 |
| Varying $Synfreq @ Nimgs = 64 |
| 50 | 85.22 | 82.00 | 83.57 |
| 100 | 84.73 | 82.73 | 84.18 |
| 200 | 85.78 | 83.39 | 84.14 |
| 400 | 85.76 | 83.19 | 84.61 |
| 1 epoch | 84.79 | 81.3 | 84.62 |

**TABLE 1. Single Device accuracies and efficiency ratios for a VGG13 model, CIFAR-10 dataset on synthetic data. Baseline accuracy for learning on real data with the same hyperparameters is 88.6%, as shown in grey.**
perform various experiments to empirically that privacy is guaranteed, and TOFU is robust to various data leakage attacks in Section IV-C. Section IV-D provides an analysis of the communication cost and computational overhead added by utilizing TOFU in federated platforms.

A. SINGLE DEVICE EXPERIMENTS

1) SETUP

We demonstrate our results on the CIFAR-10 dataset. It comprises of 50,000 training samples and 10,000 validation samples of 10 classes each. For all experiments, we use a VGG-13 [7] network. We use an SGD optimizer with learning rates 0.02, decayed by 0.2 at the 250th and 400th epochs, with a mini-batch size of 64 for a total of 500 epochs. The maximum baseline accuracy we achieve by training on real data is 88.6%. In this section all non-baseline results are shown for learning on purely synthetic data, with varying Synfreq, Nimgs and synthetic image sizes. More details are shown in Appendix C. The results are tabulated in Table 1.

2) EFFICIENCY CALCULATION

The number of parameters of each synthetic image is 3072 (32 × 32 × 3) if no prior is enforced on the image, 1024 (32 × 32 × 1) if single-channeled images are synthesized, and 768 (16 × 16 × 3) if downsampled images are synthesized. The size of each datapoint (image, label, α trio) in the synthetic dataset \( D_{\text{syn}} \) is the sum of the size of the synthetic image + 10 (number of soft labels) + 1 (α per image). Hence the total size of the synthetic dataset \( D_{\text{size}} \) is the size of each data point multiplied with Nimgs. The communication efficiency (η) is then calculated as:

\[
\eta = \frac{M_P \times C_B}{D_{\text{size}} \times C_S}
\]

where \( M_P \) is the number of model parameters that are exchanged per full weight update, and \( C_B \) and \( C_S \) are the number of baseline and synthetic communication rounds respectively.

3) VARYING NIMGS

Table 1 shows that increasing the synthetic dataset size improves accuracy, but reduces communication efficiency as we need to send more parameters per communication round. We achieve good accuracies by learning on only synthetic data, with an average accuracy drop of 3%, 5.5% and 4% for synthetic images of original size, grayscale images, and downscaled respectively across all considered Nimgs. For further experiments, we fix Nimgs = 64 as a good trade-off point. The exchange frequency for both the baseline and the synthetic case for all Nimgs is set as 200. Baseline training reaches full accuracy sooner than learning with synthetic data as expected, but even after accounting for that, we are able to achieve up to 121x more communication efficiency. The corresponding results for MNIST are provided in Appendix C-B.

4) VARYING SYNFREQ

Here we show that whether we generate synthetic images to match a gradient as often as 50 minibatches or as late as once an epoch (updates from 782 minibatches), we can converge to a reasonable accuracy. All simulations take similar number of epochs to converge, and are able to converge to very similar accuracies. In Table 1, we compare efficiency when the synthetic images are being communicated instead of the gradient, and we assume that both of these are communicated as often as the mentioned Synfreq. However, as we see in the federated setup in the next section, the frequency of communication can vary between the synthetic data and the real data. In those cases, a lower Synfreq will result in a requirement for more communication rounds and hence achieve lesser communication savings.

5) FORCING A PRIOR ON SYNTHETIC IMAGE SIZES

Here, we experiment with enforcing images to not follow the same distribution as the real dataset by constraining their size. In the first experiment, the synthetic images are constrained to have a single channel (results in column 2 of Table 1). In the second experiment, we constrain images to be half the width and height (results in column 3 of Table 1). To be compatible with expected image size before being fed into the network for gradient calculation, the grayscale images are duplicated across the 3 channels and the smaller images are upsampled to the correct size by copying the nearest neighbors pixel value. The results show that we get very low drop in accuracies from full sized images. The average accuracy drop from full sized synthetic images to images with height and width halved is only 0.5% across all experiments, and 2% to grayscale images. However, we can see that we get approximately 4x and 3x improvement in communication efficiency as a result of reducing the number of parameters to be communicated or learned per image.

6) DISCUSSION

We successfully show that synthetic data can be used to learn, with small accuracy drops for CIFAR-10, using only synthetic data. This drop is later recovered in the federated setup. For full size images, the average accuracy drop from baseline is 3% at 17x communication efficiency. For grayscale images, we get an average accuracy drop of 5% at 49x communication efficiency and 3.5% drop at 65x savings for images with width and height halved. We also wish to mention reiterate that the larger the networks, the more the
TABLE 2. Accuracy and efficiency of the federated platform on MNIST. The baseline of FedAvg is shown in grey. The best accuracy setting (using synthetic data) for each set of experiments for using only synthetic data is highlighted in bold and shown in cyan.

| Synfreq | Varying Nimgs @ Synfreq = 1 local epoch | + 3 Rounds of FedAvg |
|---------|-----------------------------------------|----------------------|
|         | Max Acc. (%) | Comm. Rnd. | Comm. Eff. | Max Acc. (%) | Comm. Eff. |
| FedAvg  | 98.91 | 456 | 1.0× | 98.06 | 6.6× |
| 32      | 95.39 | 129 | 6.9× | 98.14 | 4.1× |
| 64      | 96.06 | 104 | 4.2× | 98.01 | 3.5× |
| 96      | 96.77 | 83  | 3.6× | 98.07 | 4.1× |
| 128     | 95.23 | 53  | 4.2× | 97.91 | 3.1× |
| 25      | 92.00 | 130 | 3.4× | 98.29 | 3.3× |
| 50      | 91.71 | 56  | 7.9× | 98.19 | 7.5× |
| 1 epoch | 96.06 | 104 | 4.2× | 98.14 | 4.1× |
| 2 eps   | 95.52 | 142 | 3.1× | 97.91 | 3.1× |

savings that can be achieved with our method, since the size of updates will increase but the size of images remains constant.

B. FEDERATED LEARNING EXPERIMENTS

1) SETUP

We now discuss the federated experiments conducted on 5 and 10 clients for CIFAR-10 and MNIST, respectively, with an IID distribution of data. This results in each client having 157 minibatches of size 64 image-label pairs for CIFAR-10 and 97 minibatches of size 64 for MNIST. We compare with FedAvg [10] as our baseline, with exchanges happening once per epoch. We assume a participation rate of 1. The results are shown in Table 2 for MNIST and Table 3 for CIFAR-10.

2) EFFICIENCY CALCULATION

In the case where full gradients are exchanged at the end in order to recover the accuracy, the communication efficiency ($\eta$) is calculated by including the number of rounds of full gradient exchange ($C_F$):

$$\eta = \frac{M_P \times C_B}{(D_{size} \times C_S) + (M_P \times C_F)}$$

3) VARYING SYNFRQ AND NIMGS

We get the best results at a Synfreq equal to one local epoch for both datasets, similar to FedAvg. We note that there is no more variation in efficiency with varying Synfreq settings than single device experiments. That is because in this case, our Federated baseline is fixed at a Synfreq of 1 epoch, irrespective of what the Synfreq of synthetic images is. Increasing the synthetic dataset size (Nimgs) results in better or comparable accuracy but sends more parameters per communication round. We get best accuracy for Nimgs = 96 for both datasets, seeing a drop of 2% and 12% for MNIST and CIFAR-10, with 3.6× and 14.7× communication efficiency, respectively. We note that the drop is more than what we see in single device, since each round incurs approximations from both up and down-communications. We show that we can recover this drop almost completely in just 3 communication rounds for MNIST and 15 rounds in CIFAR-10, still allowing for 3 − 6.6× communication efficiency for MNIST and 5.4 − 8.8× for CIFAR-10, as seen in the last columns of Tables 2 and 3. Additionally, we realize that showing maximum accuracy might skew communication efficiency in our benefit, since FedAvg reaches higher accuracy, which will naturally take more communication rounds. To account for this, we also report communication efficiency at iso-accuracy in Tables 6 and 7 for MNIST and CIFAR-10, respectively in Appendix C. When we consider 72% as the baseline accuracy for CIFAR-10, our efficiency of 5.4 − 8.8× reduces to 2.4 − 4.7×. This shows that our method is still more efficient under stricter evaluation conditions. We found that the combination of Synfreq=1 epoch and Nimgs=64 provides a good trade-off point between communication efficiency and accuracy drop.

4) ENFORCING A PRIOR ON SYNTHETIC IMAGE SIZES

We also experimented with restricting the optimization to enforce a prior on the synthetic images in the federated setup (for the CIFAR-10 experiments). Table 3 shows that generating 1-channeled and downsampled images instead of 3-channeled images provided communication efficiency at the cost of accuracy. Using synthetic images with such priors provided a communication efficiency of 39.3 − 156.3×, however, with an accuracy drop of 14 − 25% from the FedAvg baseline. However, this drop can be reduced to less than 1% with 15 additional rounds of full weight update exchange, while still achieving communication efficiency of 8.6 − 10.3×.

C. EMPIRICAL EXPERIMENTS TO SHOW PRIVACY

In this section, we perform additional experiments to empirically test the privacy conferred by weighted gradients generated by synthetic images versus full gradients.

1) SANITY CHECK EXPERIMENTS

As a baseline, we first show the effect of the Inverting Gradients (IG) attack [2] on full gradients and the synthetic gradient provided by TOFU. We used the exact same setup in both situations. Figure 4 shows the empirical evidence in a situation where IG was able to recover many features of the dataset when full gradients were exchanged (Figure 4b). However, it failed to retrieve any useful information about the dataset when the TOFU framework was used (Figure 4c). These experiments were performed using Synfreq=5, Nimgs=8 for a batchsize of 8.

2) VARIATIONS OF THE IG ATTACK

It is to be noted that IG attacks (like most attacks) were designed to attack a known label. Since TOFU generates soft labels, IG cannot be directly used to retrieve the images. We show two versions of IG, one in which we use one label per class as the target, and one in which we use the predicted class of the synthetic image as the target.
TABLE 3. Accuracy and efficiency on the federated platform on CIFAR-10. The baseline of FedAvg is shown in grey. The best accuracy setting for each set of experiments using only synthetic data is highlighted in bold and shown in cyan.

| Synthetic Image Size | Using Only Synthetic Data | Additional Rounds of Full Gradient Exchange |
|----------------------|---------------------------|------------------------------------------|
|                      | Max Acc. (%) | Comm. Rounds | Comm. Eff. | + 5 Rounds | Max Acc. (%) | Comm. Eff. | + 10 Rounds | Max Acc. (%) | Comm. Eff. | + 15 Rounds | Max Acc. (%) | Comm. Eff. |
| **FEDAVG**            | 88.73        | 166          | 1.0×       |            | 83.67        | 19.1×       | 86.40        | 12.1×       | 87.29        | 8.8          | 83.60        | 12.3×       | 87.80        | 8.3×        |
| Nings                | Varying Nings @ Synfreq = 1 local epoch |
| 32                   | 67.12        | 352          | 44.9×      |            | 85.05        | 12.4×       | 87.18        | 9.0×        | 88.30        | 7.1×         | 84.30        | 10.3×       | 87.80        | 6.9×        |
| 64                   | 75.00        | 400          | 19.8×      |            | 86.50        | 10.2×       | 87.74        | 7.8×        | 88.39        | 6.3×         | 86.63        | 7.7×        | 87.40        | 5.4×        |
| 96                   | 76.02        | 359          | 14.7×      |            | 84.43        | 8.0×        | 86.84        | 6.5×        | 87.86        | 5.4×         | 84.40        | 7.1×        | 86.83        | 5.1×        |
| 128                  | 76.00        | 357          | 10.6×      |            | 84.66        | 7.8×        | 86.83        | 5.6×        | 87.86        | 5.4×         | 84.40        | 7.1×        | 86.83        | 5.1×        |
| **Synfreq**          | Varying Synfreq @ Nings = 64 |
| 50                   | 63.03        | 439          | 18.0×      |            | 84.23        | 11.7×       | 86.58        | 8.6×        | 87.69        | 6.9×         | 84.40        | 7.1×        | 86.83        | 5.4×        |
| 100                  | 70.07        | 547          | 14.5×      |            | 84.40        | 10.1×       | 86.63        | 7.7×        | 87.60        | 6.3×         | 84.40        | 7.1×        | 86.83        | 5.4×        |
| 1 epoch              | 75.00        | 400          | 19.8×      |            | 85.05        | 12.4×       | 87.18        | 9.0×        | 88.30        | 7.1×         | 84.40        | 7.1×        | 86.83        | 5.4×        |

FIGURE 4. Sanity check experiments to empirically show privacy offered by TOFU. (Rows from top to bottom): (a) Ground Truth Images from the dataset (b) Images reconstructed using IG attacks on full gradient exchanges (c) Images reconstructed using IG attacks on synthetic gradients decoded from TOFU.

In Figure 5, we show that TOFU is still able to maintain privacy regardless of how the target label is reconstructed. In Figure 5a, we reconstructed images using each class label (1-10) as a target label. In Figure 5b, we used the original target labels used to reconstruct the ground truth images. We also would like to note that if the IG attack was redesigned for soft labels and spanning ratios, the images that would be reconstructed will resemble the synthetic dataset exactly. Hence privacy would still be maintained. The setup used was similar to the one used in Figure 4c, that is, we used the label with the maximum softmax probability value as the target label to perform the IG attack.

FIGURE 5. Images reconstructed with different target labels.

FIGURE 6. Images reconstructed from TOFU when different batchsizes are used. (a) TOFU fails when the batchsize is 1. The synthetic image produced resembles the data. (b) Batchsize=8. (c) Batchsize=32. The label with maximum softmax probability was used as the target label for all the cases.
3) PRIVACY UNDER VARYING CONDITIONS
To empirically show that the synthetic gradients produced are private regardless of the hyperparameters chosen, we performed three experiments. We choose the standard hyperparameters to be batchsize = 8, Nimgs=8, and Synfreq=5 and vary each hyperparameter one at a time. First, we varied the batchsize and in this case, set Nimgs to be equal to the batch size. Figure 6 shows that TOFU fails when the batchsize is 1, but succeeds at maintaining the privacy for batchsize > 1. Next, we fixed the batchsize and Synfreq, and varied the number of synthetic images (Nimgs). Figure 7 shows that images reconstructed from the synthetic gradient still maintain privacy. Thirdly, we fix the batchsize and Nimgs to be 8 and vary the Synfreq. Figure 8 demonstrates that while privacy is maintained at all Synfreq, the obfuscation of data information increases with increase in Synfreq.

D. COMMUNICATION AND COMPUTATIONAL OVERHEADS
1) COMMUNICATION COST OF ADDITIONAL FULL GRADIENT EXCHANGES
TOFU shares full weight updates for the last few epochs to regain full accuracy, which need to be encrypted to ensure privacy. For a conservative estimate while ensuring privacy, we assume that we need to encrypt all parameters sent during all communication rounds for both methods, including synthetic data and full weight updates. Secure aggregation [6], a commonly used protocol, shows that the communication cost is $O(n + k)$ for the client and $O(nk + n^2)$ for the server, where $k$ is the dimension of the vector being encrypted and $n$ is the number of clients. Comparing the encryption cost between FedAvg and TOFU for the same number of clients reduces to a ratio of the total parameters sent. This means that encryption retains the efficiency benefits of our method. The results show that TOFU can learn both MNIST and CIFAR-10, distributed in an IID setup, with an average of $\sim 4.6 \times$ and $\sim 6.8 \times$ communication efficiency and less than an 1% average accuracy drop.

2) COMPUTATIONAL COST OF ENCODING AND DECODING IMAGES
While TOFU ensures privacy in federated learning setups, it does come at an additional computational cost. In order to quantify the overhead, we performed Algorithms 1 and 2 using an NVIDIA GeForce GTX 1060 GPU on VGG13 and the CIFAR-10 dataset for a Synfreq of 200 and a batchsize of...
32. The average time to perform these experiments is reported in Figure 9. We also provide the average time taken by a regular SGD training for the same batchsize and number of minibatches for comparison.

We found that the computational overhead added by the encoding and decoding is only affected by the size of the synthetic dataset (Nimgs) and the number of iterations of optimization (max iterations) performed. The overhead is independent of the synthesis of frequency (Synfreq) and the batchsize used. We would also like to point out, that the decoding of the synthetic dataset consists only of one forward and one backward pass of the model. In our experiments, the decoding took lesser than 10 milliseconds and the majority of the compute time (and computational overhead) was from the encoding algorithm.

V. CONCLUSION
In the standard federated learning algorithm, clients carry out local learning on their private datasets for some minibatches, and communicate their weight updates to a central server. The central server aggregates the weight updates received from all clients, and communicates this update back to all clients. There are two major bottlenecks in this procedure; it is communication inefficient and it is shown that gradient and weight updates can be inverted into the data that generated them, violating user privacy. In this work, we introduce TOFU, a federated learning framework for communication efficiency and to enhance protection against data leakage via gradients. We encode the weight updates to be communicated into a weighted summation of the gradients of a much smaller set of proxy data. The proxy data resembles noise and we empirically show that inversion from various data leakage attacks result in revealing this noise rather than user data. Additionally, data is far lower in dimensional complexity than gradients, improving communication efficiency. We also show that this proxy data can be downsampled in size from the original data that generated the target gradients without much drop in accuracy, thus providing more communication efficiency. Since proxy data only approximates gradients, we observe a small drop in accuracy when learning only from this synthetic data. We show that the accuracy can be recovered by a very few communication rounds of full weight updates. To ensure privacy in this phase, we recommend encrypting the updates. Since these rounds are very few in comparison to the number of rounds where we exchange synthetic data, we are still able to maintain communication efficiency. We show that we can learn the MNIST dataset, distributed between 10 clients and the CIFAR-10 dataset, distributed between 5 clients to accuracies comparable to FedAvg, with an average of ~ 4.6× and ~ 10.3× communication efficiency and less than an average 1% accuracy drop, respectively. Availability of more data and compute capabilities has encouraged network sizes to grow. Since input data usually is of fixed dimensions, the communication efficiency advantages of TOFU are expected to grow with network size.

ACKNOWLEDGMENT
(Manish Nagaraj and Isha Garg contributed equally to this work.)

APPENDIX A
TOFU: PSEUDO CODE
Here, we provide the pseudocode for the operation of TOFU. Algorithm 1 describes the encoding of the weight update into gradients from synthetic data. The updates and gradients are treated as a tuple, one for each layer. The goal of the encoding algorithm is to produce a synthetic dataset \( D_{syn} \) which comprises of synthetic images \( x_{syn} \), synthetic soft labels \( y_{syn} \), spanning ratios \( \alpha_{syn} \), and scaling ratios \( \gamma_{syn} \). ALgorithm 2 describes how to decode the synthetically created weight update \( U_{syn} \) and the true weight update \( U_{real} \) as shown in Step 2 of Algorithm 1. Scaling ratios are then calculated to ensure the synthetic and target updates are of the same magnitude, shown in Step 3 of Algorithm 1.

Algorithm 1: Synthetic Dataset Creation (Encode)

| Input  | Target weight update: \( U_{real} \).
|--------|----------------------------------|
|        | model weights: \( \theta \), max iterations: \( I_{max} \), learning rates for optimizer: \( \beta_x \), \( \beta_y \), \( \beta_\alpha \) |
| Output | Synthetic dataset: \( D_{syn} \) in the form of images: \( x_{syn} \), soft labels: \( y_{syn} \), spanning ratios: \( \alpha_{syn} \), scaling ratios: \( \gamma_{syn} \) |
|        | 1) Initialize \( D_{syn} = [x_{syn}, y_{syn}, \alpha_{syn}] \sim \mathbb{N}(0, I) \) |
|        | 2) For \( i = 1 \) to \( I_{max} \):
|        | a) Forward pass \( D_{syn} \) and compute gradients \( L_{syn} = \sum \alpha_i L_{crossEntropy}(\theta(x_{syn}), y_{syn}) \), \( U_{syn} = \nabla_{\theta} L_{syn} \)
|        | b) Optimize \( D_{syn} \) to minimize cosine similarity between target and synthetic update \( R_{loss} = \cos \_sim(U_{syn}, U_{real}) \)
|        | \( x_{syn} \leftarrow x_{syn} - \beta_x \_grad_{\text{syn}} \_loss \)
|        | \( y_{syn} \leftarrow y_{syn} - \beta_y \_grad_{\text{syn}} \_loss \)
|        | \( \alpha_{syn} \leftarrow \alpha_{syn} - \beta_\alpha \_grad_{\text{syn}} \_loss \)
|        | 3) Compute the scaling ratio for each layer of the model:
|        | For \( l = 1 \) to Number of layers:
|        | a) \( \gamma_{syn, l} = \frac{\|U_{real,l}\|_2}{\|U_{syn,l}\|_2} \)

Algorithm 2 describes how to decode the synthetically created dataset back into the update \( U_{syn} \), which is just the reverse process of the Encode algorithm. The synthetic loss \( L_{syn} \) is calculated by taking a weighted average of the cross-entropy loss of all the synthetic images and labels. The gradient of this loss with respect to the model parameters, scaled according to the scaling ratios creates \( U_{syn} \).

Next, we describe the application of TOFU in the federated setup in Algorithm 3. For each communication round, there are four phases as shown in the algorithm. The first phase
Algorithm 2 Recreating Weight Update (Decode)

**Input**: Model weights: $\theta$, Synthetic dataset: $D_{syn}$,
in the form of images: $x_{syn}$; 
soft labels: $y_{syn}$; spanning ratios: $\alpha_{syn}$ 
and scaling ratios: $\gamma_{syn}$

**Output**: Synthetic Weight Update: $U_{syn}$

1) Forward pass $D_{syn}$ and compute gradients

$$L_{syn} = \sum_{i} \alpha_{i} \text{CrossEntropy}(\theta(x_{syn}), y_{syn})$$

$$U' = \frac{\nabla L_{syn}}{\nabla \theta}$$

2) Scale the gradient layerwise with the scaling ratio
For $l = 1$ to Number of layers:

a) $U_{syn,l} = U_{l}^{'} * \gamma_{syn,l}$

Algorithm 3 TOFU, the Federated Setup

1) Server Initializes Global model $\theta_{glob}$
2) For round $= 1, \ldots, R_{max}$
   a) up-communication: For clients $= 1, \ldots, k$
      i) Load $\theta_{loc} = \theta_{glob}$
      ii) While batch id $< \text{synfreq}$:
         - Each client learns on local data, with
         - $\theta_{new}$ = final weight
      iii) The target weight update is the difference between the starting and the final weight,
         $U_{real} = \theta_{loc} - \theta_{new}$
   iv) Reset client weights to global weights
      $\theta_{new} = \theta_{glob}$
   v) Encode the true gradient into synthetic data and send to server:
      $D_{syn} = \text{Encode}(U_{real}, \theta_{loc}, \beta)$
   vi) The server decodes the synthetic gradient
      $U_{syn,k} = \text{Decode}(\theta_{glob}, D_{syn})$.
   b) The server aggregates synthetic updates from all clients and decides target update for
down-communication:
      $U_{serv} = \frac{1}{k} \sum_{k} (U_{syn,k})$
   c) Down-communication: server generates synthetic data to send the updated weights to all clients as well as updates its global model
      $D_{synserver} = \text{Encode}(U_{serv}, \theta_{glob}, \beta)$
      $\theta_{glob} \leftarrow \theta_{glob} - \text{Decode}(\theta_{glob}, D_{synserver})$
   d) Each client decodes the synthetic data and updates the synced weights:
      For clients $= 1, \ldots, k$:
      i) $U_{syn_{down,k}} = \text{Decode}(\theta_{glob}, D_{synserver})$
      ii) $\theta_{glob} \leftarrow \theta_{glob} - (U_{syn_{down,k}})$

FIGURE 10. Synthetic Images produced through the various communication rounds of TOFU. The rows, from top to bottom are, randomly sampled synthetic images encoding the up-communication of Client 1 at the start of the algorithm, at the 200th communication round, and at the communication round with maximum accuracy (400th communication round). This was performed on the CIFAR-10 dataset, distributed across 5 clients in an IID fashion, encoding at the end of every epoch. We see that the images do not bear any resemblance to the dataset.

is the up-communication. The clients first load their last seen global model as the local model. They each train on their local dataset to produce the target weight updated $U_{real}$. They encode this using Algorithm 1 and up-communicate this dataset to the server. The server decodes each of the clients synthetic dataset using Algorithm 2 to produce $U_{syn,k}$. The second phase is similar to that of standard federated learning where the synthetic updates from all clients are aggregated (averaged in our case). In the third, down-communication phase, the server encodes the average weight update into a synthetic dataset using a similar process described in the up-communication phase. The server also updates its own model based on this synthetic dataset to ensure all entities, clients and server, maintain the same global model at all communication rounds. The last phase, the client update phase, involves all the clients decoding the server’s encoded weight update to update their global model.

APPENDIX B

SYNTHETIC IMAGES

In this section, we aim to show how that the synthetic images do not resemble the dataset throughout the learning progress. We show results for experiments on CIFAR-10, with data distributed in an IID fashion across 5 clients. The frequency of synthesis is set to 1 local epoch and 64 images are used for both up and down-communication encoding. We have shown the encoded up-communication
TABLE 4. Single Device accuracies and the communication rounds required to reach maximum accuracy for a VGG13 model, CIFAR-10 dataset on synthetic data. The baseline accuracy for learning on real data with the same hyperparameters is 88.6%, as shown in grey. The best accuracy setting for each set of experiments using only synthetic data is highlighted in bold and shown in cyan. The network is trained with a batch-size of 64, with 782 minibatches making up an epoch. This means that the communication rounds for the baseline will vary depending on Synfreq. For instance, a Synfreq = 200 has 4 communication rounds per epoch, making the communication rounds for the baseline for that Synfreq = 158 × 4 = 632. Similarly, this number would be twice that = 1264 for Synfreq = 100.

| Single Device Accuracy for CIFAR-10, VGG13 | Synfreq | Varying Nimgs @ Synfreq = 200 |
|------------------------------------------|---------|-------------------------------|
| Max. Acc. (%) | Rnds | Max. Acc. (%) | Rnds | Max. Acc. (%) | Rnds |
| Synfreq | Varying Nimgs @ Synfreq = 200 | 32 | 84.05 | 1880 | 81.24 | 1848 | 82.61 | 1968 |
| | | 64 | 85.78 | 1652 | 82.39 | 1988 | 84.14 | 1980 |
| | | 96 | 86.05 | 1944 | 83.63 | 1932 | 85.29 | 1712 |
| | | 128 | 86.81 | 1836 | 84.09 | 1984 | 85.74 | 1960 |
| Baseline Accuracy without using synthetic data: 88.6% (158 epochs) | | | | | | | | |
| Nimgs | Varying Nimgs @ Synfreq = 64 | 50 | 85.22 | 7504 | 82.00 | 6480 | 83.57 | 6368 |
| | | 100 | 84.73 | 3794 | 82.73 | 3992 | 84.18 | 3712 |
| | | 200 | 85.78 | 1652 | 83.39 | 1988 | 84.14 | 1980 |
| | | 400 | 85.76 | 974 | 83.19 | 950 | 84.61 | 932 |
| | | 1 epoch | 84.79 | 403 | 81.3 | 441 | 84.62 | 469 |

TABLE 5. Single Device accuracies and efficiency ratios for a LeNet model, MNIST dataset on synthetic data. Baseline accuracy for learning on real data with the same hyperparameters is 99.32%, as shown in grey. We show communication rounds with corresponding communication efficiency compared to the baseline in brackets. The best accuracy setting for each set of experiments for using only synthetic data is highlighted in bold and shown in cyan.

| Single Device Accuracy for MNIST, Modified LeNet-5 | Max Acc. (%) | Comm. Rnds | Comm. Eff. |
|-----------------------------------------------|-------------|------------|------------|
| Baseline | 99.52 | 495 | 1.0× |
| Nimgs | Varying Nimgs @ Synfreq = 100 | 32 | 97.59 | 1194 | 0.8× |
| | | 64 | 97.84 | 512 | 0.9× |
| | | 96 | 97.93 | 853 | 0.4× |
| | | 128 | 98.14 | 890 | 0.3× |
| Synfreq | Varying Nimgs @ Synfreq = 64 | 50 | 97.85 | 1587 | 0.3× |
| | | 100 | 97.84 | 512 | 0.9× |
| | | 200 | 98.01 | 540 | 0.9× |
| | | 400 | 97.21 | 413 | 1.2× |

TABLE 6. Federated results for Isoaccuracy (for 92% and 98%) on MNIST. The baseline of FedAvg is shown in grey. The best efficiency setting for each set of experiments using only synthetic data is highlighted in bold and shown in cyan.

| MNIST, LeNet5, Number of Clients = 10 IID Distribution | Accuracy: 95% | Accuracy: 98% |
|--------------------------------------------------------|--------------|--------------|
| | Comm. Rnds | Comm. Eff. | Syn. Rnds | FedAvg Rnds | Comm. Eff. |
| FEDAVG | 12 | 1.0× | - | 48 | 1.0× |
| Nimgs | Varying Nimgs @ synfreq = 1 local epoch | 32 | 121 | 0.2× | 129 | 0.7× |
| | | 64 | 55 | 0.2× | 104 | 0.8× |
| | | 96 | 60 | 0.1× | 526 | 0.5× |
| | | 128 | 53 | 0.1× | 439 | 0.4× |

APPENDIX C
EXPERIMENT DETAILS AND ADDITIONAL ANALYSIS
In this section, we describe the details of the learning mechanisms we use in our experiment. We then discuss additional results that supplement the tables in the main text.

A. SETUP DETAILS
We use the same setup for both single device and federated experiments, and for both baselines and our experiments. For MNIST, we use a slightly modified LeNet-5. We use relu non linearity and maxpool. There are two parallel instantiation of the second convolutional layer (with 16 filters) with their outputs summed together. We use an SGD optimizer and a learning rate 0.1 for 200 epochs. For CIFAR-10, we use a VGG13 with an SGD optimizer and a learning rate of 0.02, with a decay of 0.2 first at 250 and then at 400 epochs for 500 epochs. All experiments were performed with a batchsize of 64. The baselines for the single device performance are...
TABLE 7. Federated results for Isoaccuracy (for 72% and 87%) on CIFAR-10 using 3 channel synthetic images. The baseline of FedAvg is shown in grey. The best efficiency setting for each set of experiments using only synthetic data is highlighted in bold and shown in cyan.

```plaintext
| CIFAR10, VGG13, Number of Clients = 5, IID Distribution | Accuracy: 72% | Accuracy: 87% |
|---------------------------------------------------------|---------------|---------------|
|                                                          | Comm Rnds | Comm Eff. | Syn Rnds | FedAvg Rnds | Comm Eff. |
| FEDAVG                                                   | 25        | 1.0×      | -        | 82          | 1.0×       |
| Nimgs                                                    | Varying 3 Channel Nimgs @ synfreq = 1 local epoch
| 64                                                      | 254       | 4.7×      | 254      | 8           | 5.0×       |
| 96                                                      | 250       | 3.2×      | 250      | 6           | 4.7×       |
| 128                                                     | 250       | 2.4×      | 250      | 10          | 3.2×       |
```

standard learning on real data, and for federated is the FedAvg algorithm. For the encoding algorithms, we use an Adam optimizer with a learning rate of 0.1 set for images, labels and αs, run for 1000 iterations.

B. SINGLE DEVICE PERFORMANCE

Table 4 shows the number of communication rounds required to reach maximum accuracy for the experiments described in IV-A. Similar to the experiments performed on the CIFAR-10 dataset, we performed experiments on the MNIST dataset and the slightly modified LeNet-5 described in the previous section. Table 5 shows the results for the experiments performed on the MNIST dataset on the slightly modified LeNet-5. We now show the learning curves to show how validation accuracy varies with communication rounds. Figures 11 and 12 show the performance of TOFU on MNIST and CIFAR-10 on a single device platforms, for different Synfreq and Nimgs. We set switch₁ to 100 communication rounds in MNIST and 200 communication rounds in CIFAR-10.

C. ISOACCURACY PERFORMANCE OF TOFU

For a fairer comparison, we also report results on iso-accuracy, since the baseline FedAvg reaches higher accuracies, which will naturally take more communication rounds and skew efficiency in our favor. Table 6 shows the number of communication rounds needed to achieve iso-accuracy with the baseline (FedAvg) for the experiments on the MNIST dataset. For 95%, only the synthetic images are sufficient, for 98%, additional full weight update rounds are needed. Table 7 shows the number of communication rounds needed to achieve iso-accuracy using TOFU for experiments on the CIFAR-10 dataset, with synthesis every local epoch. We observed, that to achieve 72%, only the synthetic images were sufficient, and to achieve 87%, 6 to 10 additional full weight update rounds were needed. In both Tables 6 and 7 ‘Syn. Rnds’ denotes the number of communication rounds using synthetic data while ‘FedAvg Rnds’ denotes the number of rounds for full weight update exchange.
**FIGURE 14.** Federated Performance of TOFU on the CIFAR-10 dataset, distributed in an IID fashion amongst 5 clients. The $switch_2$ was finetuned to provide optimal efficiency and is denoted on the graphs by a star.

**FIGURE 15.** Corresponding Validation Loss observed in the federated performance of TOFU on the CIFAR-10 dataset (IID distribution amongst 5 clients).

**D. LEARNING CURVES OF THE FEDERATED PLATFORM EXPERIMENTS**

Figure 13 shows the learning curves for the federated platform experiments using the MNIST dataset. $switch_2$ was manually fine tuned in order to achieve the highest communication efficiency. The dashed lines on the graph represent the additional full weight updates.

Figures 14 and 15 show the learning curves for the federated platform experiments using the CIFAR-10 dataset. Similar to the federated MNIST experiment, $switch_2$ was manually fine tuned in order to achieve the highest communication efficiency and is denoted by a star for each curve. The dashed lines on the graph represent the additional full weight updates.

**REFERENCES**

[1] L. Zhu, Z. Liu, and S. Han, “Deep leakage from gradients,” in Proc. Adv. Neural Inf. Process. Syst., vol. 32, 2019.

[2] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, “Inverting gradients—how easy is it to break privacy in federated learning?” in Proc. Adv. Neural Inf. Process. Syst., vol. 33, 2020, pp. 16037–16047.

[3] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov, “See through gradients: Image batch recovery via GradInversion,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2021, pp. 16332–16341.

[4] L. Fowl, J. Geiping, W. Czaja, M. Goldblum, and T. Goldstein, “Robbing the fed: Directly obtaining private data in federated learning with modified models,” 2021, arXiv:2110.13057.

[5] L. Fowl, J. Geiping, S. Reich, Y. Wen, W. Czaja, M. Goldblum, and T. Goldstein, “Decepticons: Corrupted transformers breach privacy in federated learning for language models,” arXiv preprint arXiv:2201.12675, 2022.

[6] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, “Practical secure aggregation for privacy-preserving machine learning,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur. New York, NY, USA: Association for Computing Machinery, Oct. 2017, pp. 1175–1191, doi: 10.1145/3133956.3133982.

[7] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2014, arXiv:1409.1556.

[8] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 770–778.

[9] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, “Visualizing the loss landscape of neural nets,” 2017, arXiv:1712.09913.
M. Nagaraj et al.: TOFU by Encoding Weight Updates Into Gradients From Proxy Data

[10] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, “Communication-efficient learning of deep networks from decentralized data,” in Proc. 26th Int. Conf. Artif. Intell. Statist., vol. 54, 2017, pp. 1273–1282.

[11] P. Kairouz et al., “Advances and open problems in federated learning,” 2019, arXiv:1912.04977.

[12] J. Konečný, H. Br. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, “Federated learning: Strategies for improving communication efficiency,” 2016, arXiv:1610.05492.

[13] S. Horvath, C.-Y. Ho, L. Horvath, A. N. Sahu, M. Canini, and P. Richtarik, “Natural compression for distributed deep learning.” 2019, arXiv:1905.10988.

[14] D. Basu, D. Data, C. Karakus, and S. Digvaji, “Qsparse-local-SGD: Distributed SGD with quantization, sparsification, and local computations,” 2019, arXiv:1906.02367.

[15] D. Alistarh, D. Grubic, J. Li, R. Tomikoma, and M. Vojnovic, “QSGD: Communication-efficient SGD via gradient quantization and encoding,” 2016, arXiv:1610.02132.

[16] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mirmonov, K. Talwar, and L. Zhang, “Deep learning with differential privacy,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur. New York, NY, USA: Association for Computing Machinery, Oct. 2016, pp. 308–318, doi: 10.1145/2976749.2978318.

[17] C. He, M. Annamaram, and S. Avestimehr, “Group knowledge transfer: Federated learning of large CNNs at the edge.” 2020, arXiv:2007.14513.

[18] J. Ion, K. Lee, S. Oh, and J. Ok, “Gradient inversion with generative image prior,” Proc. Adv. Neural Inf. Process. Syst., vol. 34, pp. 2998–2998, 2021.

[19] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, “SCAFFOLD: Stochastic controlled averaging for federated learning,” 2019, arXiv:1910.06378.

[20] H. Yu, R. Jin, and S. Yang, “On the linear speedup analysis of Stochastic controlled averaging for federated learning,” 2019, arXiv:1905.03817.

[21] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan, “Adaptive federated optimization,” 2020, arXiv:2003.00925.

[22] S. P. Karimireddy, M. Jaggi, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, “Mime: Mimicking centralized stochastic algorithms in federated learning,” 2020, arXiv:2008.03666.

[23] M. Duan, D. Liu, X. Chen, Y. Tan, J. Ren, L. Qiao, and L. Liang, “Astraea: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications,” in Proc. IEEE 37th Int. Conf. Comput. Design (ICCD), Nov. 2019, pp. 246–254.

[24] Y. Zhou, G. Pu, X. Ma, X. Li, and D. Wu, “Distilled one-shot federated learning.” 2020, arXiv:2009.07999.

[25] T. Wang, J.-Y. Zhu, A. Torralba, and A. A. Efros, “Dataset distillation,” 2018, arXiv:1811.10595.

[26] G. Cazanavette, T. Wang, A. Torralba, A. A. Efros, and J.-Y. Zhu, “Dataset distillation by matching training trajectories,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW), Jun. 2022, pp. 4749–4758.

[27] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in Proc. 3rd Int. Conf. Learn. Represent. (ICLR), San Diego, CA, USA, Y. Bengio and Y. LeCun, Eds., May 2015.

ISHA GARG received the B.Tech. degree in electrical engineering from BITS Pilani, India, in 2013, and the Ph.D. degree from Purdue University, in 2023, under the supervision of Prof. Kaushik Roy. She worked on memory design with AMD and Synopsys. She was with the National University of Singapore, researching efficient inference of convolutional neural networks with Prof. Massimo Ailiot. She is currently with the Video Engineering Team, Apple. Her research interests include efficient, robust, and private training and inference algorithms for deep learning and computer vision.

KAUSHIK ROY (Fellow, IEEE) received the B.Tech. degree in electronics and electrical communications engineering from IIT Kharagpur, Kharagpur, India, and the Ph.D. degree from the Electrical and Computer Engineering Department, University of Illinois at Urbana-Champaign, in 1990.

He was with the Semiconductor Process and Design Center, Texas Instruments, Dallas, where he worked on FPGA architecture development and low-power circuit design. He joined the Electrical and Computer Engineering Faculty, Purdue University, West Lafayette, IN, USA, in 1993, where he is currently the Edward G. Tiedemann Jr. Distinguished Professor. He is also the Director of the Center for Brain-Inspired Computing (C-BRIC) funded by SRC/DARPA. He has published more than 700 papers in refereed journals and conferences, holds 25 patents, and supervised more than 100 Ph.D. dissertations. He is the coauthor of two books on Low Power CMOS VLSI Design (John Wiley and McGraw Hill). His research interests include neuromorphic and emerging computing models, neuro-mimetic devices, spinntronics, device-circuit algorithm co-design for nano-scale silicon, non-silicon technologies, and low-power electronics. He received the National Science Foundation Career Development Award, in 1995; the IBM Faculty Partnership Award; the ATT/Lucent Foundation Award; the 2005 SRC Technical Excellence Award; the SRC Inventors Award; the Purdue College of Engineering Research Excellence Award; the Humboldt Research Award, in 2010; the 2010 IEEE Circuits and Systems Society Technical Achievement Award (Charles Desoer Award); the Distinguished Alumni Award from IIT Kharagpur; the Fulbright-Nehru Distinguished Chair; the DoD Vannevar Bush Faculty Fellow, from 2014 to 2019; the Semiconductor Research Corporation Aristotle Award, in 2015; the 2020 Arden Bement Jr. Award; the Highest Research Award given by Purdue University in pure and applied science and engineering; the Best Paper Awards at the 1997 International Test Conference, the 2003 IEEE Nano, the 2004 IEEE International Conference on Computer Design, and the 2006 IEEE/ACM International Symposium on Low Power Electronics and Design; the 2005 IEEE Circuits and System Society Outstanding Young Author Award (Chris Kim); the 2006 IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS Best Paper Award; the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design Best Paper Award; and the 2013 IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) Best Paper Award. He was a Purdue University Faculty Scholar, from 1998 to 2003. He was a Research Visionary Board Member of Motorola Labs, in 2002. He was the M. Gandhi Distinguished Visiting Faculty Member with IIT Bombay and the Global Foundries Visiting Chair with the National University of Singapore. He has been in the editorial board of IEEE Design and Test, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS, IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, and IEEE TRANSACTIONS ON ELECTRONIC DEVICES. He was a Guest Editor for Special Issue on Low-Power VLSI in IEEE Design and Test, in 1994, IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, in June 2000, IEEE Proceedings—Computers and Digital Techniques, in July 2002, and IEEE JOURNAL ON EMERGING AND SELECTED TOPICS IN CIRCUITS AND SYSTEMS, in 2011.

**Manish Nagaraj** (Graduate Student Member, IEEE) received the B.E. degree in electronics and communication engineering from the PES Institute of Technology, Bengaluru, India, in 2017, and the M.Sc. degree in electrical and computer engineering from Purdue University, West Lafayette, IN, USA, in 2019, where he is currently pursuing the Ph.D. degree in electrical and computer engineering. His research interests include efficient AI frameworks for edge devices and neuro-inspired computer vision algorithms.