FEDERATED DATASET DISTILLATION

A report

submitted by

M K SAI VENKAT VARUN

in partial fulfilment of the requirements

for the award of the degree of

MASTER OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

JUNE 2020

THESIS CERTIFICATE

This is to certify that the thesis titled **Federated Dataset Disitllation**, submitted by **M K SAI VENKAT VARUN**, to the Indian Institute of Technology, Madras, for the award of the degree of **Master of Technology**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Sheetal Kalyani
Research Guide
Professor
Dept. of Electrical Engineering
IIT-Madras, 600 036

Place: Chennai

Date: 15th June 2020

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and gratitude to Dr. Sheetal Kalyani for guiding my research with her ideas, thoughts, and insights.

I am very grateful to my research partner Vishnu Raj for his advice and help at each stage.

I would like to thank my friends and family for supporting and helping me through the entire process.

My sincere thanks are due to the Department of Electrical Engineering, IIT Madras for providing me with teaching, facilities, and other services.

ABSTRACT

KEYWORDS: Federated Learning, Distributed Optimization, Dataset Distillation, Deep Learning

On-device machine learning enables the training process to utilize a large amount of user-generated data samples. Numerous algorithms such as Federated Learning, Federated Distillation, and Hybrid Federated Distillation have been proposed hitherto to exploit this benefit. These are typically based on exchange of local gradients, local parameters, or even averaged outputs. In this report we study an alternative– Federated Dataset Distillation (FDD), a distributed model training algorithm which creates synthetic, distilled versions of users' datasets which are provided to users to train on. We show that using the proposed algorithm, a device can achieve good accuracy on an image classification task in a bandwidth-constrained cooperative learning scenario.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

DL	Deep Learning
FL	Federated Learning
FD	Federated Distillation
FAug	Federated Augmentation
GAN	Generative Adversarial Network
HFD	Hybrid Federated Distillation
GD	Gradient Descent
FDD	Federated Dataset Distillation
CNN	Convolutional Neural Network

NOTATION

$\mathbf{x}$	Real dataset, comprised of samples x_i
$\tilde{\mathbf{x}}$	Distilled dataset, comprised of synthetic samples $\tilde{x}_i$
M_{real}	Sample size of real dataset
M	Sample size of distilled dataset
$\tilde{\eta}$	Optimal learning rate for distilled dataset
$\ell(x_i, \theta)$	Loss of neural network with parameters θ at data point x_i
θ_0	Initial distribution of weights of neural network
N	Number of clients
$\mathcal{D}_k$	Local dataset of client
n	Size of each client's local dataset
$\tilde{\mathcal{D}}_k$	Distilled local dataset of client
$\tilde{\mathcal{D}}$	Union of clients' distilled datasets

CHAPTER 1

INTRODUCTION

Applications of Machine Learning in communications have a long history covering a wide range of applications comprising of channel modelling and prediction, localization, equalization, decoding, quantization, compression, demodulation, modulation recognition, and spectrum sensing to name a few (Ibnkahla (2000), Bkassiny *et al.* (2012), Kim *et al.* (2017), Kim *et al.* (2018), and references therein). The advent of open-source Deep Learning (DL) libraries and readily available specialized hardware along with the astonishing progress of DL in computer vision have stimulated renewed interest in the application of DL for communications and networking.

An interesting application of DL in communication is the concept of Federated Learning (Konečenỳ *et al.* (2016)). Federated Learning is a machine learning setting where a shared global model is trained from a federation of large number of clients each with unreliable and relatively slow network connections. A principle motivating example of Federated Learning is enabling mobile phones to collaboratively learn a shared prediction model while keeping all the training data on device. In this algorithm the global model is shared among the users using the model parameters.

Not long ago Hinton *et al.* (2015) proposed *network distillation* as a way to transfer the knowledge from an ensemble of many separately-trained networks into a single, typically compact network, performing a type of model compression. Taking inspiration from that phenomenal work, Wang *et al.* (2018) proposed an algorithm that performs a similar, but orthogonal task - distill a dataset. Unlike network distillation, in Dataset Distillation the model is kept fixed and the knowledge of the entire training dataset, which typically contains thousands to millions of images, is encapsulated into a small number of synthetic training images. It was shown that one can go as low as one synthetic image per category, training the same model to reach surprisingly good performance on the synthetic images. Considering Dataset Distillation as a compression algorithm, we were inspired to apply it to the concept of cooperative learning to create an algorithm similar to Federated Learning.

In this work, we suggest an alternative to Federated Learning where clients share *distilled* versions of their datasets amongst themselves instead of their model parameters with a central server. We show that when the union set of distilled data from all the clients is shared with a new client, he can have a head-start on performance for a fraction of the bandwidth that would be required to share their *complete* datasets with him.

CHAPTER 2

PRECURSORY ALGORITHMS

2.1 Federated Learning

Machine Learning algorithms often rely on distributing the optimization of their model parameters over multiple machines due to their enormous datasets and models of increasing complexity. Existing ML algorithms are designed for highly controlled environments (such as data centers) where the data is distributed among machines in a balanced and IID fashion, and high-throughput networks are available. Konečenỳ *et al.* (2016) wished to extend this application to mobile phones and tablets which contain unprecedented amounts of data. However, bearing in mind the data privacy concerns that would arise, they proposed Federated Learning (FL) as an alternative to this setting. In this algorithm, a shared global model is trained under the coordination of a central server, from a federation of participating devices. The participating devices (clients) are typically large in number and have slow or unstable internet connections. The training data is kept locally on users' mobile devices, and the devices are used as nodes performing computation on their local data in order to update a global model. This differed from conventional distributed machine learning due to the very large number of clients, highly unbalanced and non-IID data available on each client, and relatively poor network connections. We suppose that we have extremely large number of devices in the network — as many as the number of users of a given service, each of which has only a tiny fraction of the total data available. In particular, we expect the number of data points available locally to be much smaller than the number of devices. Additionally, since different users generate data with different patterns, it is reasonable to assume that no device has a representative sample of the overall distribution. Under the naive implementation of Federated Learning (FSVRG, *etc.*), each client shares their entire model with the central server. This becomes a communication bottleneck when the models used by clients are large. Hence, Konečený *et al.* followed up their research on FL with work which focused on the communication efficiency of the algorithm (Konečený

et al. (2016)). In this work they proposed two ways to reduce up-link communication costs: *structured* updates, where we directly learn an update from a restricted space parametrized using a smaller number of variables, *e.g.* either low-rank or a random mask; and *sketched* updates, where we learn a full model update and then compress it using a combination of quantization, random rotations, and subsampling before sending it to the server. With the goal of minimizing the inter-device communication overhead in FL, while still protecting the users' privacy, Jeong *et al.* (2018) proposed Federated Distillation (FD): a distributed model training algorithm whose communication payload size is much smaller than FL, particularly when the model size is large. Further stating that the user-generated data samples are likely to become non-IID across devices, which commonly degrades the performance compared to the case with an IID dataset, they proposed another algorithm: Federated Augmentation (FAug). It is a data augmentation scheme using a generative adversarial network (GAN) that is collectively trained under the trade-off between privacy leakage and communication overhead. The trained GAN empowers each device to locally reproduce the data samples of all devices, so as to make the training dataset become IID.

Results: FSVRG, the Federated Learning algorithm proposed, converged to optimal test classification accuracy in just 30 rounds of communication. It outperformed the basic algorithm of Stochastic Gradient Descent for learning the global model. In terms of modifications made for reducing the up-link costs, structured updates performed better than sketched updates. This is because by sketching one throws away some of the information obtained during training. After the modification, a modest accuracy of 85% was obtained on CIFAR10 dataset while in total communicating less than half of what would be required to upload the original data. Federated Distillation and Hybrid FD (Ahn *et al.* (2019)) performed better than classic FL in a low-SNR (-10dB), fewer channels ($T = 3000$) scenario. When tested with MNIST, HFD outperformed FD along with all other schemes when number of channels used was small ($1000 \leq T \leq 5000$). When more channels were used, FL performed the best.

Although quite a few newer, better cooperative learning algorithms have been proposed, the fundamental setup of such algorithms remains the same. Multiple clients obtain a fraction of the total dataset which they train on, and share certain results from the training to develop a global model. We too adopt the same setup for our algorithm, as one can see in Chapter 3.

2.2 Dataset Distillation

Formerly, algorithms such as ensemble learning, network distillation, and model compression (Ba and Caruana (2014), Hinton *et al.* (2015), Radosavovic *et al.* (2018)) have been studied to simplify Deep Learning. In all these algorithms the models are manipulated while the data remains constant. In their work,Wang *et al.* (2018) investigated if the dataset, which typically consists of thousands of samples, could be compressed while the model remained fixed. Their goal was to generate a synthetic dataset, much smaller than the original, which delivered the same performance as the full dataset on the same network. The algorithm (Fig. 2.1) does this by first deriving the network weights as a differentiable function of the synthetic data, and given this, instead of optimizing network weights, optimizing pixel values of these synthetic images.

Input: $p(\theta_0)$: distribution of initial weights; M: the number of distilled data
Input: α: step size; n: batch size; T: the number of optimization iterations; $\tilde{\eta}_0$: initial value for $\tilde{\eta}$
Initialize $\tilde{\mathbf{x}} = \{\tilde{x}_i\}_{i=1}^{M}$ randomly, $\tilde{\eta} \leftarrow \tilde{\eta}_0$
for each training step $t = 1$ to T **do**
  Get a minibatch of real data $\mathbf{x}_t = \{x_{t,j}\}_{j=1}^{n}$
  Sample a batch of initial weights $\theta_0^{(j)} \sim p(\theta_0)$
  for each sampled $\theta_0^{(j)}$ **do**
    Compute updated parameter with GD: $\theta_1^{(j)} = \theta_0^{(j)} - \tilde{\eta}\nabla_{\theta_0^{(j)}}\ell(\tilde{\mathbf{x}}, \theta_0^{(j)})$
    Evaluate the objective function on real data: $\mathcal{L}^{(j)} = \ell(\mathbf{x}_t, \theta_1^{(j)})$
  end for
  Update $\tilde{\mathbf{x}} \leftarrow \tilde{\mathbf{x}} - \alpha\nabla_{\tilde{\mathbf{x}}}\sum_j \mathcal{L}^{(j)}$, and $\tilde{\eta} \leftarrow \tilde{\eta} - \alpha\nabla_{\tilde{\eta}}\sum_j \mathcal{L}^{(j)}$
end for
Output: distilled data $\tilde{\mathbf{x}}$ and the optimized learning rate $\tilde{\eta}$

Figure 2.1: Algorithm of Dataset Distillation

The authors of this paper aimed to learn a tiny set of synthetic data $\tilde{\mathbf{x}} = \{\tilde{x}_i\}_{i=1}^{M}$, with $M \ll M_{real}$ (size of real dataset), and a corresponding learning rate $\tilde{\eta}$ so that a single GD step using this synthetic $\tilde{\mathbf{x}}$ boosts performance on the real dataset. Given an initial θ_0, we obtain the synthetic dataset and $\tilde{\eta}$ that minimize the objective $\mathcal{L}$:

$$(\tilde{\mathbf{x}}^*, \tilde{\eta}^*) = \arg\min_{\tilde{\mathbf{x}},\tilde{\eta}} \mathcal{L}(\tilde{\mathbf{x}}, \tilde{\eta}; \theta_0) = \arg\min_{\tilde{\mathbf{x}},\tilde{\eta}} \ell(\mathbf{x}, \theta_1) = \arg\min_{\tilde{\mathbf{x}},\tilde{\eta}} \ell(\mathbf{x}, \theta_0 - \tilde{\eta}\nabla_{\theta_0}\ell(\tilde{\mathbf{x}}, \theta_0))$$

Distilled data obtained in the above manner, from a given initialization θ_0 (called *fixed initialization*), did not generalize well to other initialization weights. This was because the result was encoding the information of both the dataset ($\mathbf{x}$) and the initialization θ_0. To address this issue, the authors made the provision to create a small set of

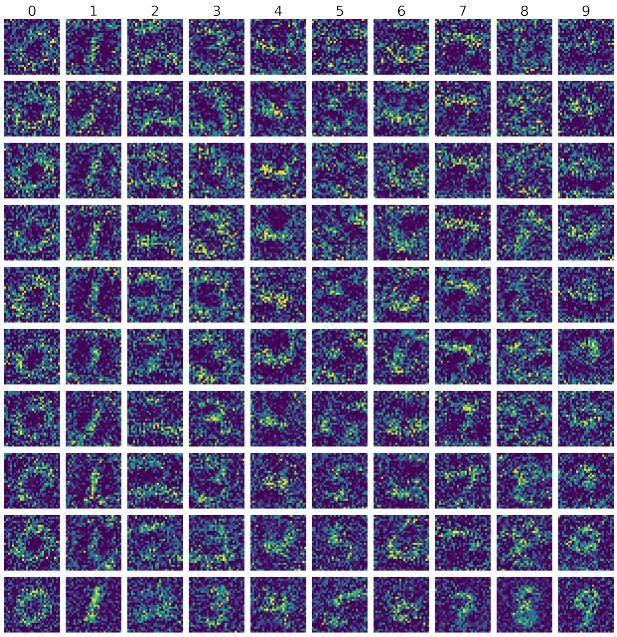
distilled data that can work with networks with random initializations from a distribution $p(\theta_0)$ (called *random initialization*). This can be seen in Step 4 of the algorithm.

The algorithm can be extended by performing GD in Step 6 to multiple sequential GD steps each on a different batch of distilled data and learning rate,*i.e.*, each step i is:

$$\theta_{i+1} = \theta_i - \tilde{\eta}_i \nabla_{\theta_i} \ell(\tilde{\mathbf{x}}_i, \theta_i)$$

and changing Step 9 to backpropogate through all steps. The recent technique of *back-gradient optimization* was used for faster and low-memory gradient calculations. It formulates the necessary second order terms into efficient Hessian-vector products (Pearlmutter (1994), Paszke *et al.* (2017)). The performance can be further improved by training the network with same distilled images for multiple epochs of the GD step(s). For each epoch, the algorithm cycles through all GD steps, where each step is associated with a different batch of distilled data. The learning rates are not tied across epochs as later epochs often use smaller learning rates.

Results: The authors' experiments showed that using the algorithm, the distilled dataset of MNIST can boost the performance of LENET network up to 94% with fixed initialization, and up to 79.5% with random initialization. The size of the distilled dataset in these experiments was $M = 10$ and $M = 100$ (Fig. 2.2a) respectively. Similarly the distilled dataset (of size $M = 100$) of CIFAR10 (Fig. 2.2b) boosted the performance of Alex Krizhevsky's network (Krizhevsky (2012)) (accuracy 80% when fully trained) to 54% with fixed initialization, and 36.8% with random initialization. It was also observed that increasing the number of GD steps significantly improved the performance. A similar but slower trend was seen in number of epochs. It was also seen that longer training time helped the model learn all the knowledge from the distilled images, but the performance was limited by the total number of images.



(a) 100 MNIST images (79.5% ± 8.1%)

(b) 100 CIFAR10 images (36.8% ± 1.2%)

Figure 2.2: Experiment results of dataset distillation with expected test accuracy when trained on network with unknown random initialization

CHAPTER 3

FEDERATED DATASET DISTILLATION

3.1 Algorithm

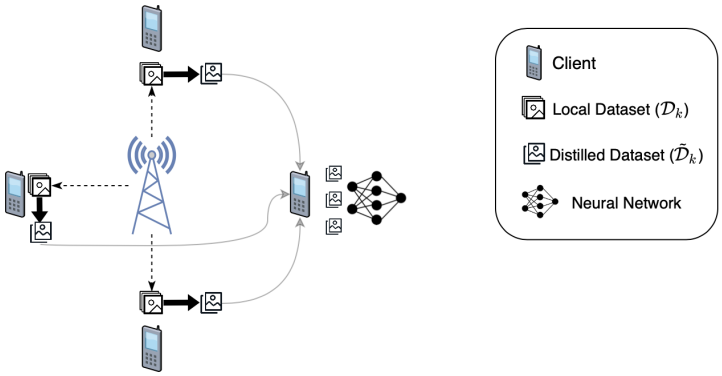


Figure 3.1: Schematic Overview of Federated Dataset Distillation (FDD)

In Federated Dataset Distillation (FDD) we consider a small group of clients who hold subsets of a larger dataset. Similar to the context of FL seen previously, this data might be what a client could gather from a larger dataset - images, cellular communication data, and so on. However, in contrast to FL, here the size of the subset is significantly larger than the number of clients ($n \gg N$). As illustrated in Fig. 3.1, we consider a wireless edge learning system with N users (clients), each consisting of a neural network and a unique local dataset $\mathcal{D}_k$. To enable cooperative learning, the devices are allowed to wirelessly communicate with each other (or perhaps a common Access Point). Each user distills his dataset $\mathcal{D}_k$ (of M_{real} samples) to a new, synthetic dataset $\tilde{\mathcal{D}}_k$ (of size $M \ll M_{real}$) using the Dataset Distillation Algorithm with appropriate hyper-parameters. Each user transfers these smaller datasets which are then integrated and used to train a new user to the dataset.

3.2 Evaluation

We conducted an experiment where N clients train a Convolutional Neural Network (CNN) to carry out image classification based on subsets of the MNIST and CIFAR10

Algorithm 1 Federated Dataset Distillation

procedure DISTILLATION PHASE

for User $k = 1, 2, \ldots, N$ **do**

Distill dataset $\mathcal{D}_k = (\mathbf{x}_k, \mathbf{y})$ to $\tilde{\mathcal{D}}_k = (\tilde{\mathbf{x}}_k, \mathbf{y})$

return $\tilde{\mathcal{D}}_k$

end for

procedure TRAINING PHASE

Combine distilled datasets received: $\tilde{\mathcal{D}} = \bigcup_{k=1}^{N} \tilde{\mathcal{D}}_k = (\{\tilde{\mathbf{x}}_k\}_{k=1}^{N}, \mathbf{y})$

Train a new user on this new dataset $\tilde{\mathcal{D}}$

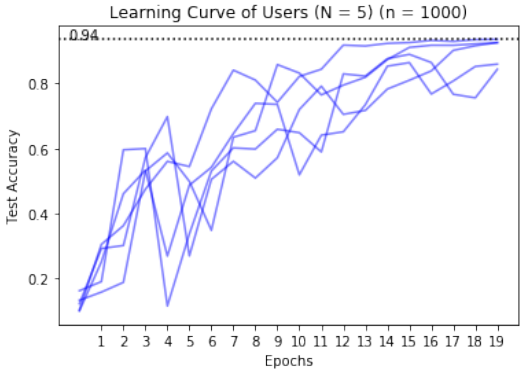
datasets available at their device. We randomly select disjoint sets of n samples from the training dataset, and allocate each set to a device. The MNIST and CIFAR10 samples are trained on LENET and Krizhevsky's neural net respectively. We first observe what performance each user can achieve if they were to train on their local dataset alone. Then, each user distills their dataset, and all N such distilled datasets are merged to form a custom dataset. This is given to a new $(N+1)^{\text{th}}$ user who has a network similar to the others. We observe how his performance evolves when trained on this custom dataset and tested on the entire test dataset.

MNIST

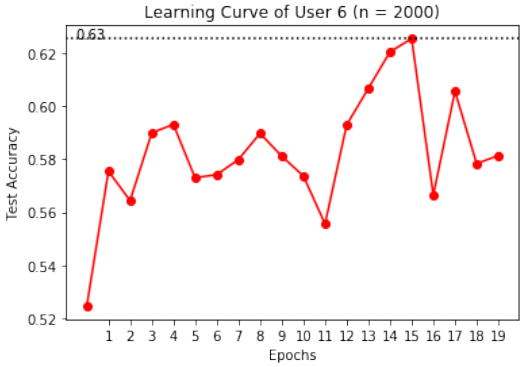
The experiment was performed with $N = 5$ users and subsets of sizes $n = 1000$ samples and $n = 5000$ samples.

When the networks are trained on 1000 samples, the accuracy maxes out around 94% at 20 epochs (Fig. 3.2a). The datasets are then distilled for 40 epochs consisting of 2 epochs (repetitions) of 20 steps each (configured as (40,20,2)). The sixth user's learning curve also ascends, although in a rather irregular fashion and only up to 63%, till 15 epochs and then begins to fall (Fig. 3.2b). One might find this behaviour inconsistent since the sixth user's dataset is larger ($20 \times 2 \times 5$ users $\times 10$ samples$= 2000$ samples) than that of any other user. However, the problem here is not the *size* of the dataset, but the *information* within. Hence, it is within reason to believe that decreasing accuracy is due to over-fitting.

For $n = 5000$ samples, as expected, the learning curves reach a higher saturation



(a) Learning Curves of N Users



(b) Learning Curve of $(N+1)^{\text{th}}$ User

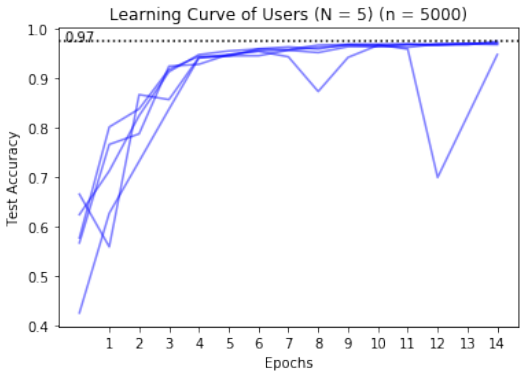
Figure 3.2: $n = 1000$ MNIST samples for $N = 5$ users distilled for (40,20,2) to give 2000 synthetic samples

(97%) than the previous case (Fig. 3.3a). The sixth user also achieves a higher maximum accuracy (76%) for the same degree of distillation (Fig. 3.3b) when compared to $n = 1000$.

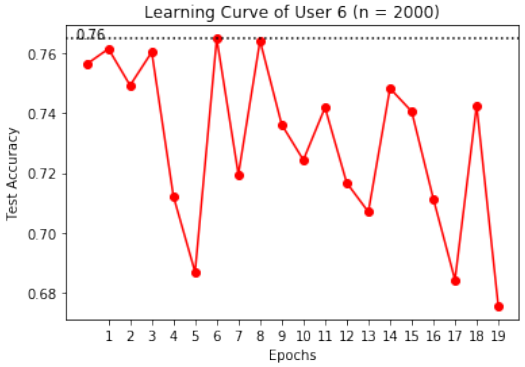
The above reasoning for why the sixth user's learning curve descends after a point can be verified from Figure 3.4. Here, as the number of epochs are increased, the information contained in the distilled data increases thereby allowing higher maximum accuracy (86%) and delaying the point at which over-fitting begins.

CIFAR10

Out of the 50,000 training samples of the original dataset, 10,000 (largest disjoint subsets) were given to each user. The users achieved a maximum accuracy of 67% when trained, for 35 epochs, on this data alone (Fig. 3.5a). The distilled data gave the sixth user an accuracy of 40% when trained for one epoch. The accuracy declined for every training epoch thereafter (Fig. 3.5b).



(a) Learning Curves of N Users



(b) Learning Curve of $(N+1)^{\text{th}}$ User

Figure 3.3: $n = 5000$ MNIST samples for $N = 5$ users distilled for (40,20,2) to give 2000 synthetic samples

3.3 Inferences

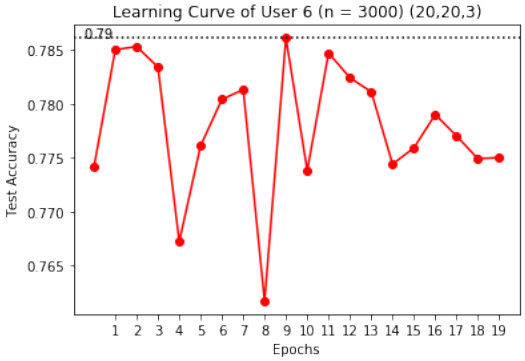
The following inferences were made from the results of the experiment:

1. The most trivial observation one can make from the results is that the performance of the new user is much better if he was given the real data. To understand how better he can really perform, we compared the learning curves of the user with the union of the users' real datasets and the union of their distilled counterparts. As seen in Fig. 3.6a the difference in accuracy is around 20%. However, for one-tenth of the bandwidth, this accuracy can be considered fairly good.

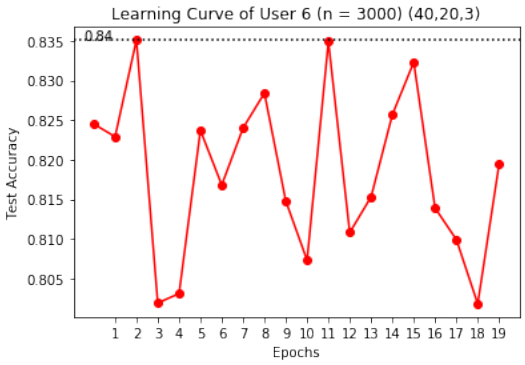
2. It can be seen that the new user's performance increases with the number of samples of distilled data provided to him. To verify this, we measured his performance while varying the number of steps of distillation (which linearly varies with the number of samples). It was observed (Fig. 3.6b) that the user has higher accuracy with larger number of samples.

3. In the previous section it was pointed out that with more time to distill, the algorithm transfers more information from the real dataset into the distilled images (see Fig. 3.4). This was verified (Fig. 3.6c) by gradually increasing the number of epochs of distillation while keeping the number of samples produced constant.

4. The lack of variance in the learning curves of distilled data could perhaps be attributed to the design of the data. The intended purpose of this data was to



(a) Learning Curve for (20,20,3) distillation

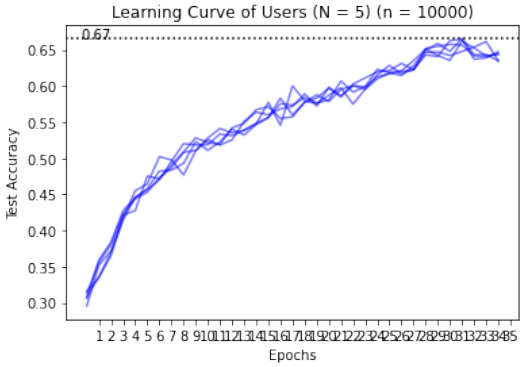


(b) Learning Curve for (40,20,3) distillation

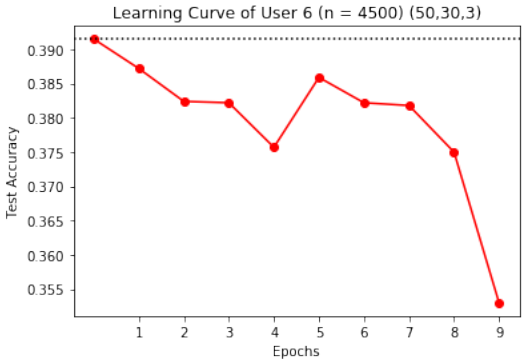
Figure 3.4: $n = 5000$ MNIST samples for $N = 5$ users distilled with two different configurations

provide good performance when trained for just one step of GD. Hence, multiple steps of optimization should not make a large difference to their performance.

5. The distilled dataset can be considered as a sufficient statistic of the real data for the given neural network. The dataset $\tilde{\mathcal{D}}$ can be viewed as a statistic of the real dataset $\mathcal{D}$ which contains all the information needed to compute any estimate of the parameters (of the network).

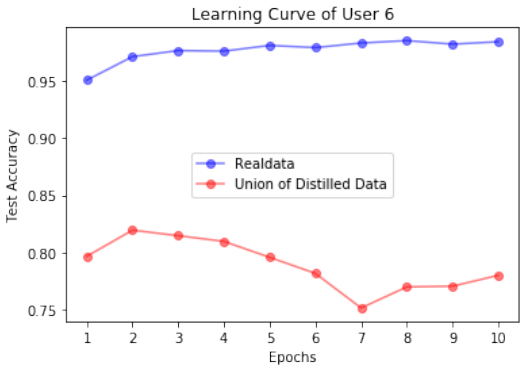


(a) Learning Curves of Users

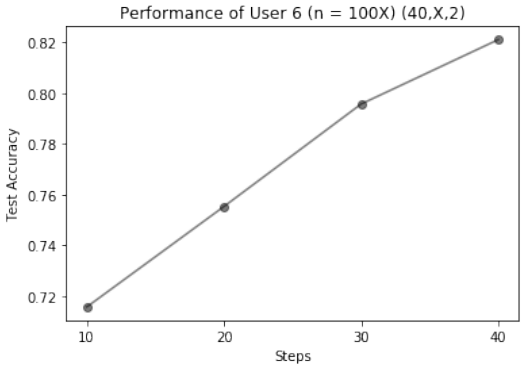


(b) Learning Curve of $(N+1)^{\text{th}}$ User

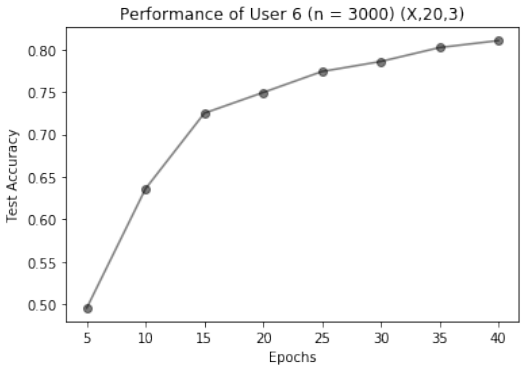
Figure 3.5: $n = 10,000$ CIFAR10 samples for $N = 5$ users distilled for (50,30,3) to give 4500 synthetic samples



(a) Real vs. distilled data



(b) Effect of no. of samples



(c) Effect of time

Figure 3.6: Inferences verified with $N = 5$ users with $n = 5000$ samples (MNIST) each

CHAPTER 4

CONCLUSIONS

In this work, we proposed an alternative to Federated Learning that does not require a central server to process data; one where the data of the users is shared amongst themselves. Using the algorithm of Dataset Distillation, we showed that for a fraction ($\approx 10\%$) of bandwidth required to share the users' original data, one could achieve good accuracy for a new client in the system. However, concluding that the approach is stunningly superior to existing methods would not be completely fair nor correct

Another fast-growing application of DL in communication is the interpretation of the communication system as an auto-encoder, and to think of its design as an end-to-end reconstruction task that seeks to jointly optimize the transmitter and receiver components (O'Shea and Hoydis (2017)). This has been proven to be superior to the traditional communication system design because a DL-based system can capture the non-linear, non-stationary imperfections of the channel often left out by classic signal processing algorithms. Moreover, it has been observed that optimizing the blocks, that constitute the traditional communication system, independently is not necessarily better than an end-to-end optimized system.

In future work, we wish to apply the technique discussed in this work to a practical system. A suitable application would be in the improvement of the end-to-end learning of communication systems. For instance, clients who have trained on the data received at different points in a wireless network can distill and share this data. When a new client processes the union of such data from all the users, he could have a head-start on his performance (a lower BER) when he encounters the said network for the first time.

REFERENCES

1. **Ahn, J.-H.**, **O. Simeone**, and **J. Kang**, Wireless federated distillation for distributed edge learning with heterogeneous data. *In 2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 2019.

2. **Ba, J.** and **R. Caruana**, Do deep nets really need to be deep? *In Advances in neural information processing systems*. 2014.

3. **Bkassiny, M.**, **Y. Li**, and **S. K. Jayaweera** (2012). A survey on machine-learning techniques in cognitive radios. *IEEE Communications Surveys & Tutorials*, **15**(3), 1136–1159.

4. **Hinton, G.**, **O. Vinyals**, and **J. Dean**, Distilling the knowledge in a neural network. *In NIPS Deep Learning and Representation Learning Workshop*. 2015.

5. **Ibnkahla, M.** (2000). Applications of neural networks to digital communications–a survey. *Signal processing*, **80**(7), 1185–1215.

6. **Jeong, E.**, **S. Oh**, **H. Kim**, **J. Park**, **M. Bennis**, and **S.-L. Kim**, Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *In NIPS Workshop MLPCD Montréal*. 2018.

7. **Kim, M.**, **N.-I. Kim**, **W. Lee**, and **D.-H. Cho** (2018). Deep learning-aided scma. *IEEE Communications Letters*, **22**(4), 720–723.

8. **Kim, M.**, **W. Lee**, and **D.-H. Cho** (2017). A novel papr reduction scheme for ofdm system based on deep learning. *IEEE Communications Letters*, **22**(3), 510–513.

9. **Konečenỳ, J.**, **H. B. McMahan**, **D. Ramage**, and **P. Richtárik** (2016). Federated optimization: Distributed machine learning for on-device intelligence. URL `https://arxiv.org/pdf/1610.02527`.

10. **Konečený, J.**, **H. B. McMahan**, **F. X. Yu**, **P. Richtarik**, **A. T. Suresh**, and **D. Bacon**, Federated learning: Strategies for improving communication efficiency. *In NIPS Workshop on Private Multi-Party Machine Learning*. 2016. URL `https://arxiv.org/abs/1610.05492`.

11. **Krizhevsky, A.** (2012). cuda-convnet2: High-performance c++/cuda implementation of convolutional neural networks. URL `https://github.com/akrizhevsky/cuda-convnet2`.

12. **O'Shea, T.** and **J. Hoydis** (2017). An introduction to deep learning for the physical layer. *IEEE Transactions on Cognitive Communications and Networking*, **3**(4), 563–575.

13. **Paszke, A.**, **S. Gross**, **S. Chintala**, **G. Chanan**, **E. Yang**, **Z. DeVito**, **Z. Lin**, **A. Desmaison**, **L. Antiga**, and **A. Lerer**, Automatic differentiation in pytorch. *In NIPS Autodiff workshop*. 2017.

14. **Pearlmutter, B. A.** (1994). Fast exact multiplication by the hessian. *Neural computation*, **6**(1), 147–160.

15. **Radosavovic, I.**, **P. Dollár**, **R. Girshick**, **G. Gkioxari**, and **K. He**, Data distillation: Towards omni-supervised learning. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

16. **Wang, T.**, **J.-Y. Zhu**, **A. Torralba**, and **A. A. Efros** (2018). Dataset distillation. *arXiv preprint arXiv:1811.10959*.