Unsupervised Federated Learning as a Way of Capturing Disjoint Data Distributions

STANLEY WU* and KRISH SHARMA*, Khoury College - Northeastern University, USA

Our project looks to combine two techniques used in machine learning: federated or distributed learning and unsupervised learning. The necessity of this intersection lies in the practical applications of machine learning in the real world. Concretely, model training in a centralized environment where all data can be accessed by a single host is a gross simplification of the problem in industry - specifically regarding the scale of the amount of data and the policies around data sharing.

CCS Concepts: • **Machine learning**; • **Unsupervised Learning**; • **Federated Learning**;

Additional Key Words and Phrases: neural networks, federated learning, unsupervised learning, data distributions

ACM Reference Format:
Stanley Wu and Krish Sharma. 2022. Unsupervised Federated Learning as a Way of Capturing Disjoint Data Distributions. 1, 1 (November 2022), 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Typically, clients and customers do not want a company that is providing them a service to share data between customers. For example, Customer A would generally not be comfortable with their data entering Customer B's environment/deployment, especially for an on-premises deployment. However, as the provider company, one would be able to gain access to the data itself. Since data is king, it's imperative to still derive value from the production grade data being seen in customer deployments. Furthermore, it is useful for the customers by being able to have a more generalized model that captures a wider distribution of potential inputs. One approach to solving this problem is to use a generative model. The reason for this is that generative models aim to find a distribution that best captures the distribution of the data. As a result, we would be able to quantize or capture the distribution of the traits of the data by training an auto-encoder, particularly variational auto-encoder, on the data seen in each silo-ed environment. With the individual distributions collected, we would then be able to combine the multiple learned distributions or embeddings for variational and vanilla auto-encoders respectively to produce a global view of production data distribution.

In our work, we look to combine the concepts of generative modeling with federated learning as a way of learning a global data distribution over the data with stratified client-side distributions. Concretely, we allow clients to train a model on a single class with the objective of minimizing reconstruction as a means of learning a latent variable representation of the class. The parameters across all the clients are the aggregated into a centralized model which is then evaluated with out training to see how well the combined representations generalize to the full evaluation set.

*Both authors contributed equally to this research.

Authors' address: Stanley Wu, wu.sta@northeastern.edu; Krish Sharma, sharma.kri@northeastern.edu, Khoury College - Northeastern University, P.O. Box 1212, Boston, Massachusetts, USA, 43017-6221.

Manuscript submitted to ACM

The authors contributed equally to this work in both code and the writing report. We wrote the model code together as well as the evaluation and plotting code. We collaborated on the paper by writing alternating questions and taking responsibility for different figures and tables.

Our code can be found at: GitHub

1.1 Related Work

The concept of federated learning originates from the idea that mobile devices house large amounts of very user-targeted data that can be used to greatly improve any model's performance [1]. However this large amount of data is often extremely sensitive to privacy and confidentiality, which is further emphasized by the White House releasing a report in 2012 that encourages *focused collection or data minimization*[4]. However with the federated learning approach, we are able to send local model updates trained on local datasets without needing to send a potentially private dataset to the server. By isolating a privacy attack on the personal device itself rather than also on a centralized server, federated learning is able to take advantage of rich local datasets while also minimizing risk of privacy leakage[4]. McMahan et al. also focus on a **Non-IID** scenario where data distributions of local datasets are not representative of population distribution. While the work in that paper focuses mainly on supervised machine learning problems, we focus on an unsupervised variational auto-encoder application of federated learning.

Federated learning has also shown to be effective when data has been vertically divided, run on overcomplete auto-encoders, before the latent data is then aggregated for model training. This application of federated learning with auto-encoders is able to learn vertically incomplete datasets, and similar to the original intention of auto-encoders, is a practical and safe solution when building models while also protecting datasets [3].

While federated learning has been shown to be effective in the supervised setting as well as an auto-encoder setting, it is unclear how well it performs when data distributions are not shared between sessions of federated auto-encoder training. Our approach aims to show that even on disjoint data distributions, federated auto-encoders are able to pick up on each sub-auto-encoder through both an aggregated approach as well as a generative approach.

1.2 Background Information

In this paper, we focus on two core machine learning concepts: federated learning, and variational auto-encoders. Similar to regular auto-encoders, variational auto-encoders can be founded on the topic of generative models. Whereas regular auto-encoders encode the input into latent space and directly proceed on decoding the input reconstruction, variational auto-encoders encode the input into a latent *distribution*, and sample from that distribution for our latent representation before decoding. In doing so, we introduce KL divergence loss in addition to standard MSE reconstruction loss in order to train a good regularized encoding scheme [5].

Federated learning is an approach on combining the results of smaller locally trained updates on machine learning models before aggregating with the full machine learning model. Among the many aggregation methods, a simple way of thinking aggregation is the averaging of each locally trained networked parameters.

2 PROPOSED APPROACH

The key component is to train individual autoencoders on each sub-class of a dataset, aggregate the parameters of each sub-model into a single model for each dataset, and see how our approach varies in success. We then subsequently

need to validate that the reconstruction loss for the images that we will be training on is low enough that the learned parameters are meaningful and useful in isolation prior to using them for downstream generalization.

The motivation for using images is that they have a wide range of potential learnable latent features which could be embedded. This means that the federation or aggregation is non-trivial since there would need to be a way to ensure that the specific attributes learned by each individual network are similar, which poses an interesting research problem and formulation of the reconstruction loss to encourage the network to learn certain features.

Upon each of the individual networks being trained, the aggregation step would consist of combining the latent representations. In particular, the goal of aggregation would be to provide a more generalized embedding/distribution. As such we use an aggregation method of parameter merging: average the networks' parameters into one global model using the average value for each parameter.

Note that for parameter merging, we have an additional constraint that the exact same architecture has to be used such that we can do an average across all of the values that parameter takes on across the trained networks. In order to keep the experimentation consistent, we chose to only use a single architecture across our datasets and experiments.

Finally, we aim to test this for multiple datasets to see how the original distribution of the data across silo-ed environments can impact the globally learned model.

3 EXPERIMENTS

Our experiment was designed in the following manner across the datasets that we used. These datasets are described further below. As previously mentioned, in the federated design of our project we have a single coordinating party which serves as the aggregator of the weights, referred to as the server, which interfaces with each of the models that are being trained on a single class which are referred to as clients.

In order to support this design we used the Flower Python library which provides us with convenient ways of interfacing between the server and clients [2].

In each experiment, we first start up the central server with a list of classes that the centralized model should be evaluated on. Subsequently, we start each of the clients with a corresponding class that they will see at training and evaluation time. Additionally, we give each of the clients the number of epochs that will be used for each iteration of training as well as the dimensionality of the latent vectors which corresponds to the number of latent variables learned. We keep the number of epochs and the latent space dimensionality consistent across datasets.

The notion of "training iterations" is not common in traditional, centralized machine learning as epochs are used as the means of number of training iterations. In the context of federated learning, it is important to account for temporal updates for the clients which drives the motivation of multiple training iterations over time. Since this would be a new axis of variability in the case of our experimentation, we opted to not introduce temporal differences in the data that is seen. Nevertheless, we still have 3 total iterations to see how the performance of the centralized model changes as the client models become further tuned for their particular dataset. Some potential implications of multiple training iterations is that when clients are only seeing a single class, as often is the case for federated learning given the motivation of the field being "data originating from a single mobile device", they overfit to that class - this can have impacts on the averaging aggregation strategy if the parameters begin to diverge between clients and averaging them leads to poor generalization to both clients.

End to end, the server is told all of the classes that will be seen across all of the clients (but not which client has which class), the model architecture, and the dimensionality of the latent space. Then for each class that we are interested in for that dataset, we start a client for that specific class with the same model architecture, latent space dimensionality

Keep Right	Stop	Roadwork	Speed Limit	No Passing	Yield	No Entry	Bumpy Road	Pedestrians	Ahead Only
0.0031	0.0049	0.0036	0.0079	0.0026	0.0028	0.0029	0.0025	0.0059	0.0041

Table 1. GTSRB Final Client Evaluation Losses

Ankle Boot	Sneaker	Coat	Bag	Pullover	Sandal	Dress	Shirt	Trouser
0.0023	0.0017	0.0022	0.0030	0.0024	0.0029	0.0020	0.0023	0.0016

Table 2. FMNIST Final Client Evaluation Loss

Ship	Car	Cat	Truck	Bird	Frog	Deer	Horse	Dog	Plane
0.0032	0.0040	0.0036	0.0041	0.0031	0.0028	0.0027	0.0039	0.0038	0.0033

Table 3. CIFAR-10 Final Client Evaluation Loss

and training epochs. The server waits until the same number of clients have connected as there are classes that it expects and then prompts each of the clients to go through a training iteration. Each of these clients then trains for 15 epochs on their class and generates performance metrics. Upon the completion of training, the clients send an array of their weights to the server. Once the server receives the weights from all of the clients, it defines its own model by averaging all of the arrays and setting those weights as its own and runs inference on all of the classes seen across all of the clients as a means of evaluating the performance on the global dataset. This evaluation serves as a metric for how well a centralized distribution can be learned through averaging individual, disjoint distributions. We repeat this process 2 more times except the weights are not reset on the client side between iterations.

We used the following datasets for evaluating the performance of our system through the process described above:

(1) FMNIST - Images of articles of clothing
(2) GTSRB - German traffic signs
(3) CIFAR-10 - 10 class assorted image dataset

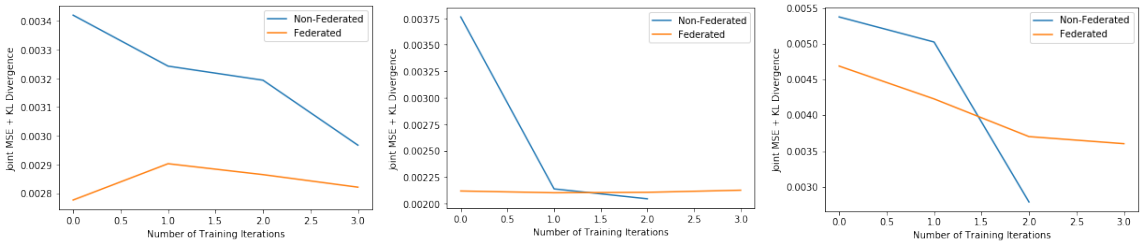
In order to evaluate performance, we use a common loss function for variational autoencoders which combines a traditional autoencoder's optimization with a loss term used for generative models. Concretely, we use a joint loss function of reconstruction loss represented by mean squared error and Kullback-Leiber Divergence. The first term evaluates how well the model has learned the data representation and the second controls the shape of the distribution of the variables to ensure that there is a parameterized Gaussian for each latent variable.

3.1 Results

We collected our results in multiple forms across clients, servers, datasets, and data types. We describe our results separated by dataset.

For our most uniform dataset of the three was the GTSRB dataset which is composed of signs. As such, we anticipated that the features learned would be similar for each of the classes as signs share the same dimensions of variability (i.e. shape, annotations, etc). We found that the clients individually perform quite well in learning the data distributions and are able to create images with low reconstruction loss as shown in the final test error tables below.

Through the client results we see that the architecture was well suited for our application since each of the clients, regardless of dataset, performed extremely well on their corresponding class on the joint loss function of MSE and KL



(a) GTSRB (b) CIFAR-10 (c) FMNIST

Fig. 1. Centralized performance vs. Non-federated approach on all 3 datasets. Performance seems to vary over training iterations for federated case. We stop non-federated approach once improvements slow, but do not adopt this approach for federated learning due to volatility of aggregation.

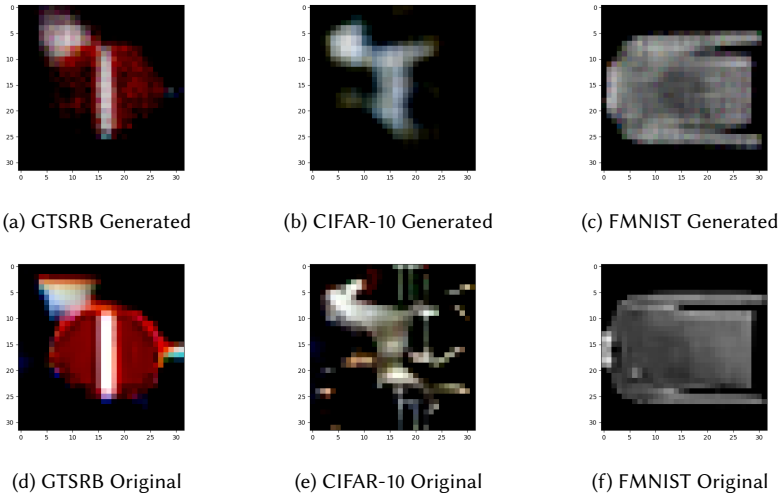
divergence. In order to compare the performance over time, we plotted the losses on the evaluation set for each of these clients which can be found in our repository. Evaluation was performed using the MSE + KL divergence on a hold out set of images of the corresponding class, we additionally sampled from these models for qualitative assessment in Figure 2. While it did not significantly impact the results, an interesting artifact that we noticed was that the test performance of the CIFAR-10 clients marginally degraded over training iterations. Overall, we see that for a simpler dataset such as GTSRB where the data is traffic signs that have relatively basic features with respect to geometry and color, the loss is slight lower than a diverse dataset such as CIFAR-10 which has complex objects such as animals and vehicles.

The focus of our work was on the aggregated, centralized models that resulted from the combination of the individual clients. As such, we performed evaluation on the centralized model that resulted from averaging the parameters from each of the client models for each dataset. These results seek to answer core questions about this experiment: what performance can we achieve on a global test set by leveraging the training done by separate models on individual classes of that global test set?

We found that our current approach generalizes reasonably well to the overall test set but is a bit unstable. Concretely, as seen in Figure 1, the loss is comparable to that of the individual clients at any given point but varies quite significantly between training iterations compared to a centralized, single model approach. One hypothesis that we would want to investigate given this artifact is whether the value of corresponding weights diverges between clients when there is a spike in the centralized test loss. Our intuition is that along the solution plane, there may be gradients that are moving in separate directions which causes the averaging of the weight values to be suboptimal with respect to the global dataset since the parameter value would be suboptimal for all of the classes.

4 CONCLUSION

In summary, through this work we aimed to uncover how individual generative models can be combined in order to learn a global distribution of the data together while only seeing a single class individually. We completed this experiment through standardizing our experimental design across three datasets where we had a single client train on an individual class. We found that using this approach, we were able to perform extremely well on an individual client basis through quantitative metrics as well as by sampling images from the learned latent distributions. However, this did not generalize to the centralized model. We found that the performance of the centralized models suffers through the averaging aggregation approach.



(a) GTSRB Generated (b) CIFAR-10 Generated (c) FMNIST Generated

(d) GTSRB Original (e) CIFAR-10 Original (f) FMNIST Original

Fig. 2. Sample generated images from the test set of each dataset from the centralized method.

The limitations of our study come from both design and the variables that we chose to keep consistent. Concretely, we standardized our experiment across multiple datasets rather than varying the approach we took on a single dataset. In hindsight, it could be valuable to find an optimal strategy for a single dataset in order to determine a better aggregation strategy (i.e. loss based weighted averaging for the centralized weights), training iterations, number of epochs, and so forth. In our approach, we leaned on the side of generalization of our experimental results which were relatively consistent across the datasets. Some of the other limitations come from the amount of compute we had available - for example, it would be much more interesting to see how this experiment design would perform on a dataset like CIFAR-100 where the number classes is much larger and represents much more variability.

In our initial design for this project, we had planned for a separate aggregation strategy of ensembling for our performance metrics but realized that this was not realistic for our project's motivation. Concretely, ensembling makes sense in a context with limited number of classes and while this is true for our data, the conclusions from this experiment would no longer generalize. Recall that the motivation for the area of federated learning was for aggregating the information learned from data on mobile devices, it would not be feasible to have a single model for every mobile device which is what an ensembling approach would require in a federated context.

REFERENCES

[1] 2017. Federated learning: Collaborative machine learning without centralized training data. https://ai.googleblog.com/2017/04/federated-learning-collaborative.html

[2] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, and Nicholas D Lane. 2020. Flower: A Friendly Federated Learning Research Framework. *arXiv preprint arXiv:2007.14390* (2020).

[3] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Xu Liu, and Bingsheng He. 2019. A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection. *CoRR* abs/1907.09693 (2019). arXiv:1907.09693 http://arxiv.org/abs/1907.09693

[4] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. 2016. Federated Learning of Deep Networks using Model Averaging. *CoRR* abs/1602.05629 (2016). arXiv:1602.05629 http://arxiv.org/abs/1602.05629

[5] Joseph Rocca. 2021. Understanding variational autoencoders (VAES). https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73