Mobile devices and the immense amount and variety of data they generate are key enablers of machine learning (ML)-based applications. Traditional ML techniques have shifted toward new paradigms such as federated learning (FL) and split learning (SL) to improve the protection of user’s data privacy. However, SL often relies on server(s) located in the edge or cloud to train computationally-heavy parts of an ML model to avoid draining the limited resource on client devices, potentially resulting in exposure of device data to such third parties.

This work proposes an alternative approach to train computationally heavy ML models in user’s devices themselves, where corresponding device data resides. Specifically, we focus on GANs (generative adversarial networks) and leverage their network architecture to preserve data privacy. We train the discriminative part of a GAN on user’s devices with their data, whereas the generative model is trained remotely (e.g., server) for which there is no need to access device true data. Moreover, our approach ensures that the computational load of training the discriminative model is shared among user’s devices – proportional to their computation capabilities – by means of SL. We implement our proposed collaborative training scheme of a computationally-heavy GAN model in simulated resource-constrained devices. The results show that our system preserves data privacy, keeps a short training time, and yields the same model accuracy as when the model is trained on devices with unconstrained resources (e.g., cloud). Our code can be found at https://github.com/YukariSonz/FSL-GAN.

KEYWORDS
federated learning, GAN, split learning

ACM Reference Format:
Pranvera Kortoçi, Yilei Liang, Pengyuan Zhou, Lik-Hang Lee, Abbas Mehrabi, Pan Hui, Sasu Tarkoma, and Jon Crowcroft. 2022. Federated Split GANs. In 1st ACM Workshop on Data Privacy and Federated Learning Technologies for Mobile Edge Network (FedEdge ’22), October 17, 2022, Sydney, NSW, Australia. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3556557.3557953

1 INTRODUCTION
The use of mobile and ubiquitous devices such as smartphones and other handheld devices equipped with a wide range of sensors has given rise to a massive amount of collected data. The data consists of both sensor data, which relates to phenomena around us, as well as data related to the individuals carrying such sensors. Many applications – such as those in the field of smart cities, smart transportation, manufacturing, supply chain, healthcare, and security – are built upon such data. The vast proliferation of application domains results in a huge amount of data being collected.
Traditionally such data is stored in the cloud for further processing and analysis. However, such centralized data analytics in the cloud, for instance, poses several data privacy issues [18]. Specifically, the cloud would have access to users’ privacy-sensitive data, which could potentially disclose their daily paths, home location, or even financial-related information. As such, fully-centralized data collection and analysis does not provide needed data privacy and security. By contrast to such traditional AI techniques, distributed collaborative learning solutions such as federated learning (FL) and split learning (SL) provide data privacy [20]. Moreover, SL is particularly suitable for training computationally-heavy learning models on resource-constrained devices. As such, FL and SL combined [11] enable a whole range of data-sensitive applications in medicine, finance, and extended reality (XR) on resource-constrained devices.

XR applications are now operating in different domains such as industry 4.0 [5], smart transportation [21], education [4], and needless to say, gaming, to create immersive and interactive experiences. Most of these applications have stringent delay requirements and collect sensitive data related to user privacy, such as real-time eye tracking and context aware sensing. As such, keeping the processing of data at local devices have great potential for enabling these applications. However, XR applications are notably computationally-heavy, relying mostly on deep neural networks to learn data models, which restricts their applicability in mobile devices due to their constrained resources.

In this article we envision a system that combines both FL and SL to enable training of generative adversarial networks (GANs) [6] with resource-constrained devices. A GAN comprises a generative model \( G \) that captures data distribution and a discriminative model \( D \) that judges if the sample data came from the raw user data or was generated by \( G \). GANs outperform traditional ML algorithms in feature learning and representation [2] and have been widely adopted for computer vision tasks which are at the core of XR applications. However, GAN algorithms are computationally-heavy, which greatly hinders their training with resource-constrained devices. Moreover, GAN can potentially raise the privacy leakage risk [13]. To address such issues, we use FL to ensure data privacy by having clients train a learning model on their own data locally, without sharing such data among clients and a remote server. In addition, we complement FL with SL to distribute the training workload among user devices. As such, each device trains only a subset of layers of the split network model [19].

Our work builds on the premise that there is an ever-growing proliferation of XR applications and services, which will become even more relevant due to technological advancements on 5G and beyond networks. As such, we need to provide dependable and resource-efficient methods to train computationally-heavy models on resource-constrained devices, with a focus on data privacy. Specifically, our contributions in this work are threefold as follows.

1. We apply FL and SL to the discriminator of a GAN to split a training model into portions, and assign them across multiple devices per client (if available) to achieve more granular task assignment than related works.
2. We devise a heuristic device selection method that turns device heterogeneity into an advantage by assigning heavier model portions to devices whose computational capabilities support it.
3. We run simulations on a controlled environment by emulating resource-constrained devices and their computational capabilities. The results show that introducing device capability-awareness along with granular task assignment reduces the training time of a discriminator, while achieving a high model accuracy.

The rest of the paper is structured as follows. Section 2 discusses related works and emphasizes the motivation of our work. Section 3 introduces our system model and formulates the problem, whereas Section 4 details the architecture of our proposed FSL-GAN solution. Section 5 evaluates the performance of our FL and SL method applied to GANs. Finally, Section 6 concludes the work with some final remarks.

## 2 RELATED WORKS

GANs [6] are a subset of machine learning models used in semi-supervised and unsupervised learning. GANs are successfully deployed in artificial intelligence (AI) applications such as medicine, computer vision, vocal AI, autonomous driving, and natural language processing [8, 9, 12, 15]. Both the generator \( G \) and the discriminator \( D \) of a GAN are deep neural networks; in particular, \( D \) is a classifier that aims at differentiating between real data stored on-premises (on the device) and fake data generated by \( G \). As such, it is highly desirable, and in some cases necessary, to train \( D \) on the devices that generate and host the real (true) data. However, such devices come with different computational capabilities which might hinder the on-site training of GAN models. Enabling distributed learning of GAN models, especially for XR applications, requires to tackle two main challenges: (i) constrained resources (i.e., heterogeneity) on the devices where we train the model, and (ii) data privacy.

Medical applications, for instance, are built upon highly privacy-sensitive patient data, and thus any ML model run on medical data must ensure its privacy. The work in [3] proposes a distributed asynchronous discriminator GAN (Asy-DGAN) framework where each data source (e.g., device) trains its discriminator \( D \) with its own local data. As such, it provides data privacy and security due to data isolation...
among devices and the generator. Similarly, [17] proposes FedGAN, a distributed framework to train a GAN across different data sources and corresponding devices. FedGAN trains both the $D$ and $G$ locally, while an intermediary entity averages the $D$ and $G$ parameters and broadcasts them back to the devices. While this approach ensures communication efficiency and data privacy, it unnecessarily trains $G$ on the devices, exhausting their resources. The work in [7] presents MD-GAN, a multi-discriminator GAN architecture that leverages FL to distributedly train an ML model over the data sets of different workers. MD-GAN assumes a single generator hosted and trained by a server (contrary to [17]) to mitigate the computational load on the workers.

FL and SL come to the rescue by ensuring both data and model privacy, while accounting for heterogeneous client capabilities. FL provides data privacy by having each client train an ML model locally, with no data sharing among participating clients. However, constrained resources of participating clients greatly hinder the applicability of FL [22], where clients might not be able to train an entire model. By contrast, SL addresses the limitations of FL by splitting the network architecture among participating clients and, possibly, a coordinating entity such as a server. For instance, [10] proposes a privacy-sensitive parallel SL method where each client trains a portion of a model with a predefined size of mini batch that reflects the size of its (local) data set. Similarly, [1] proposes a combination scheme of FL and SL to train recurrent neural networks on distributed sequential data. This way, not only the clients have access to their own sequence of sequential data, ensuring data data privacy, but they also share the burden of training a model by splitting it, and thus ensuring model privacy.

Our work, similar to existing research, successfully deploys FL and SL to ensure both data and model privacy. In addition, our proposed method optimally splits the network architecture of a model into smaller portions to be trained by individual devices, in proportion to their on-board computation capabilities. As such, our method assures that a model is trained within a given time budget by splitting a model in accordance with the computational budget of single devices.

3 BACKGROUND

3.1 System Architecture

Our proposed distributed method of training a GAN via FL and SL is shown in Figure 1. The generator model is trained by a server that is usually located in the cloud. We assume that there are no constraints in terms of computational capabilities of the server. Moreover, we relieve the computational burden on the resource-constrained devices by outsourcing the training of the generator to a third party, while we still maintain data security. In fact, the generator generates fake data, which are not privacy-sensitive. By contrast, each client trains a discriminator network with their own local data, which is not shared among them or other entities. This approach resembles that of FL. Moreover, we assume that a client can have (i.e., own) more than one device. As such, the discriminator can be trained collaboratively among such devices, where each device trains only a split (portion) of the GAN, proportional to their on-board capabilities. This approach resembles the architecture of SL. We refer to our system architecture FSL-GAN, denoting a combined use of FL and SL approaches to train GANs.

Figure 1 shows how a centralized generator shares the generated images with the local discriminators, who in turn discriminate (compare) such images to their real, true data (images) and classify them as either fake or real. The parameters of the local discriminator models are then averaged in the same manner as FedAVG [14].

3.2 System Model

The system includes the set $C = \{c_1, c_2, \ldots, c_M\}$ of $M = |C|$ FL clients, where each $c_m$ client has a set $D = \{d_1^m, d_2^m, \ldots, d_{|D|^m}^m\}$ of $N = |D|$ SL devices, and a central server. The generator model is trained by a server that is usually located in the cloud. We assume the central generator shares the generated images with the local discriminators, who in turn discriminate (compare) such images to their real, true data and classify them as either fake or real. In our architecture, each client trains a discriminator network with their own data. The client’s data resides on the client’s devices only, and thus is not shared among different clients or other entities. As such, the architecture of each discriminator network is similar to that of FL.

Training all the layers of the discriminator network model in one single device might not be feasible due to its limited resources in terms of memory, computational capabilities, and energy. As such, we opt to train a discriminator model among different devices of a given client. That is, computationally-heavy networks such as GANs are split among the devices of a client, where each device trains a portion, i.e., subset of layers, of the network. The split is done in accordance with the computational budget of single devices.
with the heterogeneous capabilities of single devices. Specifically, we consider two parameters, namely Time_Factor and Client_Capacity, to factor in the computation capability of the devices, and thus the time it takes to train a unit of network model, as well as the memory available on the devices.

4 DEVICE SELECTION IN FSL-GAN

A successful deployment of FSL-GAN relies on the condition that a client can train a GAN among their devices. As such, we take into account the device capabilities when splitting the model and assigning the corresponding portions to them. Our client selection follows two different approaches, as follows.

- **Random selection.** A client picks a device from their pool of available devices at random and assigns a model training task (e.g., portion of the model) to it. A client can choose to either assign a
  - single portion of the model to the device, and continue by randomly selecting another device from the pool until the entire model is trained; or assign
  - multiple portions of the model to the same device as long as it can train them and store the corresponding results in the local memory. Similarly, the process continues until the entire model is trained. A device is removed from the list of available devices if it cannot train any portion. Moreover, if a client does not have sufficient devices to train the entire model of a discriminator, it is removed from the list of clients that participate in FL.

- **Sort_By_Time(efficiency) selection.** We incorporate the notions of on-board memory and local processing capabilities of a device – Client_Capacity and Time_Factor, accordingly – into a parameter called efficiency. Such a parameter expresses the capacity of a device to train a model. We then sort the devices in decreasing order of efficiency, and start assigning them portions of the network to train. Similar to the method mentioned earlier, we distinguish between assigning
  - single, or
  - multiple portions of the model to a single device.

In addition, we compare our implementation of FSL-GAN to a baseline scenario in which there is only one discriminator in the system. Contrary to a Deep Convolutional Generative Adversarial Network (DCGAN) [16], which also consists of a single discriminator, we consider a baseline scenario where the discriminator splits the model and assigns portions of such a model to their devices according to the Sort_By_Time(efficiency) selection method.

5 EVALUATION

We evaluate the performance of our proposed FSL-GAN by measuring the time required for each epoch under different user selection methods (see Section 4), and the benefits of using multiple discriminators at the same time. We compare the loss of the generator of our proposed scheme with that of a baseline scenario.

**Experimental Setup.** We deployed our code on Google Colab Pro\(^1\) equipped with Nvidia Tesla P100-PCIE-16GB GPU, and run all the models sequentially on the same GPU. To simulate the scenario of FL and SL in a heterogeneous environment, we set a parameter called Time_Factor to simulate devices with different training speeds, and Client_Capacity to simulate whether the device has enough resources (e.g., memory) to store the model. We use DCGAN [16] with 3 convolution layer blocks and the MNIST data set\(^2\). At each epoch, each client will sample 24 batches of images (with BATCH_SIZE = 256) from the data set, i.e., 6144 images for each client for one epoch. For the time benchmark, we enumerate 5 clients with 4 devices each with different capacities and processing power. We measure the computation time on the slowest client, which represents the bottleneck of the whole system. Note that at this point we ignore the communication overhead from clients to central generator, and only focus on the LAN communication overhead by different splitting strategies. For simplicity, we model the LAN communication time to 50 ms for each LAN communication among devices of a single client.

For the accuracy benchmark, we measure the generator loss for 500 epochs by using a different number of discriminators, and the quality of images generated by the generator. By our evaluation we want to answer the following questions: (i) what is the trade-off between time and device capacity under different model splitting strategies? and (ii) can the generator learn under multiple discriminator environments?

**Time Benchmark.**

This benchmark allows us to compare the time consumption under different model splitting strategies. As mentioned earlier, we identified the slowest client as the system bottleneck and focus on its time consumption. Figure 2 shows the time expenditure of the slowest discriminator in the system over all epochs for different splitting strategies. We figure reports the time values averaged over all epochs along with the related standard deviations as error bars. We can observe that sorting the devices by the capacity first and assigning multiple portions to a single device, when possible, outperforms other splitting strategies. Moreover, we notice that selecting a device at random and assigning multiple parts

---

\(^1\)https://colab.research.google.com/

\(^2\)http://yann.lecun.com/exdb/mnist/
to it yields the worst performance. This is due to the fact that devices with high available memory but low computing performance exist in our simulated environment. This is typical of old devices with high memory but lack of compatibility on latest instruction sets such as AVX, AVX2, or simply without GPU.

Accuracy Benchmark.

Our accuracy benchmark allows us to evaluate whether the generator could converge under multiple discriminators instead of one. Figure 3 shows the loss of the generator for a varying number of discriminators (1, 3, 5, 7, and 8) in the system. We notice that despite the fact that all the cases have concurrent fluctuations, a higher number of discriminators helps the generator to minimize its loss. Figure 4 depicts a set of images generated by the generator in a system with different number of discriminators as a function of the number of epochs. The results show that a higher number of discriminators does not yield to a significantly faster convergence time; however, a higher number of discriminators enriches the heterogeneity of the data distribution, which is a fundamental characteristic of FL.

6 CONCLUSIONS

FL and SL bring forth new opportunities to ensure data and model privacy. As such, it enables a whole multitude of applications that are built upon privacy-sensitive data. The proliferation of these applications is such that it often requires training of computationally-heavy models such as GANs. In this article, we envision a combination of both FL and SL that enables the training of GANs in resource-constrained devices. The proposed solution consists of collaboratively training the discriminator model of a GAN among different devices of a single client by fully leveraging their heterogeneous resources. In fact, we propose a heuristic device selection method that reflects their capabilities and helps us achieve better system performance. Specifically, we show that by increasing the number of discriminators in the system we achieve a better convergence of the model, and thus achieve a lower generator loss. Moreover, this method also decreases the training time.

Future research work will further improve our device selection method such that we (i) minimize the communication overhead among devices, (ii) consider the trade-off between memory size and computational performance (e.g., the bandwidth and the number of cores on GPU/CPU), (iii) eliminate the slowest discriminator in the system, and more broadly (iv) investigate the effect of data heterogeneity in the model convergence.
REFERENCES

[1] A. Abedi and S. S. Khan. FedSL: Federated Split Learning on Distributed Sequential Data in Recurrent Neural Networks. arXiv preprint arXiv:2011.03180, 2020.

[2] Y.-J. Cao, L.-L. Jia, Y.-X. Chen, N. Lin, C. Yang, B. Zhang, Z. Liu, X.-X. Li, and H.-H. Dai. Recent advances of generative adversarial networks in computer vision. IEEE Access, 7:14985–15006, 2018.

[3] Q. Chang, Z. Yan, L. Baskaran, H. Qu, Y. Zhang, T. Zhang, S. Zhang, and D. N. Metaxas. Multi-modal AsynDGAN: Learn From Distributed Medical Image Data without Sharing Private Information. arXiv preprint arXiv:2012.08604, 2020.

[4] S. Doolani, C. Wessels, V. Kanal, C. Sevastopoulos, A. Jaiswal, H. Nambappan, and F. Makedon. A review of extended reality (xr) technologies for manufacturing training. Technologies, 8(4):77, 2020.

[5] M. Gattullo, G. W. Scurati, M. Fiorentino, A. E. Uva, F. Ferrise, and M. Bordegoni. Towards augmented reality manuals for industry 4.0: A methodology. Robotics and Computer-Integrated Manufacturing, 56:276–286, 2019.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. Communications of the ACM, 63(11):139–144, 2020.

[7] C. Hardy, E. Le Merrer, and B. Sericola. Md-gan: Multi-discriminator generative adversarial networks for distributed datasets. In 2019 IEEE international parallel and distributed processing symposium (IPDPS), pages 866–877. IEEE, 2019.

[8] C.-C. Hsu, H.-T. Hwang, Y.-C. Wu, Y. Tsao, and H.-M. Wang. Voice conversion from unaligned corpora using variational autoencoding wasserstein generative adversarial networks. arXiv preprint arXiv:1704.00849, 2017.

[9] A. Jabbar, X. Li, and B. Omar. A survey on generative adversarial networks: Variants, applications, and training. ACM Computing Surveys (CSUR), 54(8):1–49, 2021.

[10] J. Jeon and J. Kim. Privacy-sensitive parallel split learning. In 2020 International Conference on Information Networking (ICOIN), pages 7–9. IEEE, 2020.

[11] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al. Advances and open problems in federated learning. arXiv preprint arXiv:1912.04977, 2019.

[12] K. Lin, D. Li, X. He, Z. Zhang, and M.-T. Sun. Adversarial ranking for language generation. Advances in neural information processing systems, 30, 2017.

[13] Y. Liu, J. Peng, J. James, and Y. Wu. FPGA: Privacy-preserving generative adversarial network. In 2019 IEEE 25th international conference on parallel and distributed systems (ICPADS), pages 985–989. IEEE, 2019.

[14] B. McMahan, E. Moore, D. Ramage, S. Hamspon, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In Artificial intelligence and statistics, pages 1273–1282. PMLR, 2017.

[15] O. Mogren. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. arXiv preprint arXiv:1611.09004, 2016.

[16] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434, 2015.

[17] M. Rassouli, T. Sun, and R. Rajagopalan. Fedgan: Federated generative adversarial networks for distributed data. arXiv preprint arXiv:2006.07228, 2020.

[18] A. Singh and K. Chatterjee. Cloud security issues and challenges: A survey. Journal of Network and Computer Applications, 79:88–115, 2017.

[19] C. Thapa, M. A. P. Chamikara, and S. Camtepe. Splitfied: When federated learning meets split learning. arXiv preprint arXiv:2004.12088, 2020.

[20] V. Turina, Z. Zhang, F. Esposito, and I. Matta. Combining split and federated architectures for efficiency and privacy in deep learning. In Proceedings of the 16th International Conference on emerging Networking Experiments and Technologies, pages 562–563, 2020.

[21] P. Zhou, T. Braud, A. Zavodovski, Z. Liu, X. Chen, P. Hui, and J. Kangasharju. Edge-facilitated augmented vision in vehicle-to-everything networks. IEEE Transactions on Vehicular Technology, 69(10):12187–12201, 2020.

[22] P. Zhou, H. Xu, L. Lee, P. Fang, and P. Hui. Are you left out? an efficient and fair federated learning for personalized profiles on wearable devices of inferior networking conditions. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT/UbiComp), 6(3), 2022.