Privacy-Preserving GANs - Federated Learning and Fully Homomorphic Encryption Inference

Octavian-Alexandru Trifan*
Faculty Of Mathematics and Computer Science
Babes-Bolyai University
Cluj-Napoca, Romania
octavian.trifan@stud.ubbcluj.ro
0009-0005-3607-2705

Eduard-Timotei Pauliuc*
Faculty Of Mathematics and Computer Science
Babes-Bolyai University
Cluj-Napoca, Romania
eduard.pauliuc@stud.ubbcluj.ro
0009-0007-7450-0824

***Abstract*—In the face of tech companies and governments challenging the fundamental right of privacy, new innovations arise that balance the perks of Artificial Intelligence while preserving privacy. We tackle the privacy problem from two perspectives - training and inference. Regarding training, we use Federated Learning in order to show that large GAN models can be trained securely among multiple participants, with negligible performance loss. We employ Differential Privacy as another layer of security during training. For the inference, we prove the possibility of building Generative Artificial Intelligence algorithms that are capable of being privacy-preserving by using Fully Homomorphic Encryption. We build a MNIST GAN using Concrete ML and we optimize our model through Quantization Aware Training and Pruning. Lastly, we deploy on AWS and observe that the fastest inference time in a real client-server environment is 12 minutes, five orders of magnitude slower than non-FHE generation.**

***Index Terms*—Privacy, Generative Adversarial Networks, Federated Learning, Fully Homomorphic Encryption**

I. Introduction

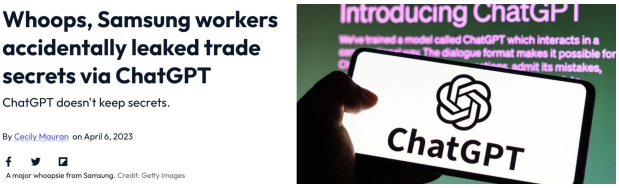


Fig. 1: "ChatGPT doesn't keep secrets" [1]

Generative AI has witnessed exponential growth in usage across many industries, especially in the last year. We are seeing AI helping us write code, writing novels, composing music, creating digital art, all from simple prompts. Generative AI is definitely altering the landscape of human-computer interaction. But can we **trust** it?

One of the questions that pop up more and more in the Generative AI space is privacy. What happens to all of our data? Are we just mindlessly feeding our personal data to these AIs, just hoping that "maybe" they respect our privacy? The harsh truth is that our right to privacy is being constantly challenged by big tech companies and governments, which are using our data in any way they might see fit.

We can look at the privacy problem of Generative AI from two major perspectives: **training** and **inference**.

First, Generative models are trained on extensive data sets, which can often contain sensitive data that the training machine can see. A solution to this problem is using a technology called "Federated Learning," (FL) which leverages the possibility of training models in a distributed manner, locally, without sending sensitive data to a central machine. While this research is not the first to explore FL outcomes, it stands out by focusing on a large model which has shown real-world practicality [2] [3], while most of the models used in other works are only experimental. An additional original contribution was the extension of the FL framework employed, NVIDIA FLARE. At the time of this work, no public implementations for training GAN models within this framework were identified.

Second, if we want to perform inference, anything we send to the server is revealed in order to perform computations. A novel way of performing computations on encrypted data is called "Fully Homomorphic Encryption", (FHE) which we will use to preserve privacy. The server will never find out what input it was given or what it has generated.

To the authors' knowledge, this is the first implementation of FHE inference for any generative algorithms.

In this paper, we have chosen Generative Adversarial Networks as they are very popular among the Generative AI algorithms. We will implement FL and FHE in order to prove that a complete flow of privacy is possible when working with GANs - from training to inference.

The need for ethical AI is growing and it needs a way to ensure privacy. This privacy is key to ensuring that the powers of Generative AI are harnessed responsibly and that we are not jeopardizing individual rights or societal norms. Technological advancement must be adopted in balance with ethical responsibilities.

II. Background knowledge

The following subsections provide an introduction into GANs, as well as FL and FHE, which we used in our approach.

*Equal contribution

A. Generative Adversarial Networks

The first formal introduction of the GAN training loop is accredited to Ian Goodfellow [4] using two Deep Neural Networks [5]. The proposed GAN architecture consists of two models, G, the generator, and D, the discriminator. The G take as input a random noise vector z and learns a mapping $G : z \rightarrow y$, where y is an output image. D takes as input an image and outputs a scalar value, where $D(x)$ signifies the probability of x being a real image, not generated by G.

In his paper, Goodfellow, when presenting future work capabilities, includes the possibility of creating a conditional GAN, by providing an additional input c to both the G and D models, where $G : \{c, z\} \rightarrow y$ and $D : \{c, x\} \rightarrow p$, where p is a probability scalar. In an unregulated generative model, the modes of generated data are not specifically controlled. Yet, when extra information is used to condition the model, as demonstrated by Mehdi Mirza in 2014 [6], the data generation process can be guided.

D is trained to correctly classify both the training samples and instances generated by G. In parallel, G is trained to minimize the value of $\log(1-D(G(z)))$, which encourages G to produce samples that D is more likely to classify as real.

The objective of the classical conditional GAN, where G tries to minimize the objective and the adversarial D tries to maximize it, is formulated as

$$\begin{aligned}\min_G \max_D \mathcal{L}_{cGAN}(G, D) =& \mathbb{E}_{c,x}[\log D(c, x)] + \\ & \mathbb{E}_{c,z}[\log(1 - D(c, G(c, z)))]\end{aligned} \tag{1}$$

Goodfellow specifies that during training, it is better, showing improved starting gradients, to train G to maximize $\log D(c, G(c, z))$. This optimization is used in most implementations of conditional GANs.

B. Federated Learning

In many practical cases, due to the necessity of large data sets in training generative models, entities do not possess enough data to train advanced GANs, especially in complex use cases. Collaboration between multiple entities that collect similar data can result in a better model than any entity would be able to create with its own data.

Data anonymization has been shown to be insufficient as a privacy solution [7]. A different approach was proposed by H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson and Blaise Agüera y Arcas at Google in 2016 [8]. In their research, they explored a method that leverages user mobile devices to train models for tasks like image classification and language modeling. This involved utilizing user photos and their text inputs, all while avoiding the centralization of this sensitive data.

In the proposed algorithm, Federated Averaging (FedAvg), each client computes multiple steps of local Stochastic Gradient Descent (i.e., performs local updates) over their data, and then sends these locally updated model parameters to the aggregator server. The server averages these models, often using a weighted average where the weights are proportional to the number of local samples. Because each client performs multiple steps of SGD, the amount of communication between server and clients is greatly reduced compared to other distributed learning algorithms.

C. Fully Homomorphic Encryption

Fully Homomorphic Encryption (FHE) is a groundbreaking concept in cryptography that allows computations to be performed on encrypted data without the need to first decrypt it. This revolutionary property makes it possible to maintain the confidentiality of the data during computations, opening new possibilities for secure data processing and privacy-preserving applications in cloud computing, machine learning, etc.

FHE supports addition and multiplication operations: $Enc(pk, m1) + Enc(pk, m2) = Enc(pk, m1 + m2)$ and $Enc(pk, m1) * Enc(pk, m2) = Enc(pk, m1 * m2)$ by leveraging large polynomials. Although a fascinating technology, FHE is a very costly and slow operation.

The advent of FHE starts in 1978 with [9] by Rivest, Adleman, and Dertouzos (the "R" and "A" in RSA). This is the first paper that defines and explores the concept of privacy-preserving computations using homomorphic encryption. However, until 2009, only Partial and Somewhat Homomorphic Encryption schemes were developed, which limited the number of operations that could be done before the noise grew too big, and the ciphertext could not be correctly decrypted. After 30 years of doubt, Craig Gentry of Stanford University finally theoretically proved the possibility of FHE in his Ph.D. thesis in 2009 [10] by introducing the concept of bootstrapping. Bootstrapping allows FHE to perform as many arithmetic operations as needed without worrying about the noise each operation introduces.

Since Gentry's breakthrough in 2009, many FHE schemes have emerged. Out of them, the most important are GSW [11] and TFHE [12], which focus on a fast bootstrapping approach, and CKKS [13], which is a leveled approach. In this paper, we will be using the framework Concrete ML [14] by Zama, which is based on TFHE. For Quantization Aware Training, a concept we will present later, we will use the Brevitas [15] library for PyTorch. As the theoretical aspects of FHE are too vast and intricate to be presented in detail here, we recommend [12] for a deep dive into the theory.

III. Related work

A. Training GANs with FL

Chenyou Fan and Ping Liu have shown that GANs can be trained in federated scenarios and produce qualitative models [16]. They have found that synchronization of both Generator and Discriminator produces the best results (Figure 2). They have used a DCGAN architecture model, with the input being a class label (a digit for MNIST or category label for CIFAR-10). In their paper, they recommend synchronizing both G and D models when the communication costs are not an issue, and synchronizing only G otherwise.

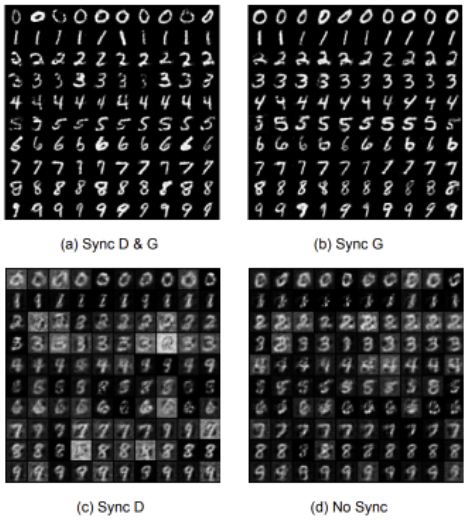


Fig. 2: Comparison between different synchronization choices [16].

The developers of the NVIDIA FLARE framework have studied the possibilities of Gradient Inversion Attacks on FL systems [17]. They provide ways in which the amount of information leaked in such attacks can be visualized and measured. Using these tools, they have found that increasing the amount of local iteration a client performs and a more diverse dataset decreases the amount of information that can be extracted in such an attack. This motivated the choice of an increased number of local iterations per training round in our experiments.

Researchers at Apple have also looked into ways to use FL concepts and apply them at scale. One of their publications [18] shows how they managed to train a Speaker Identification model using the FedAvg algorithm with different types of Differential Privacy implementations. Their work validates the utility and effectiveness of this technology in large applications.

B. FHE in Machine Learning

The 2023 paper [19] from the Zama team is the closest in concept to what we are trying to achieve in this paper. They are showing how to construct Deep Neural Networks that are compatible with FHE, and they discuss the architectural decisions behind their models.

They are also concerned with the MNIST dataset, but for a computer vision task, meaning that they are concerned with the classification of handwritten digits. They are using Brevitas for the quantization and L1-norm unstructured pruning to optimize the models. These settings, combined with Concrete-ML, give us a close representation of what we have implemented, but from a classification view instead of a generative one.

The models tested are convolutional neural networks (CNN), some fully connected (FC), with 3 or 6 layers. We can see in Table I that the inference times are inversely-proportionate to the accuracy. The hardware tests were performed on an Intel i7-11800H CPU with 8 cores and 16 threads. However, it is safe to assume that a model that keeps the accuracy at a high level will take over 300 seconds to be classified with FHE.

TABLE I: Experimental results obtained with Brevitas and Concrete-ML [19]

Network	Quant. bits	Data-set	Circuit bit width	Accuracy	Inference time (s)
3-layer FCNN	2 / 2	MNIST	6	92.2%	31
3-layer FCNN	2 / 2	MNIST	7	96.5%	77
3-layer FCNN	2 / 2	MNIST	8	97.1%	300
LeNet	2 / 2	MNIST	8	97.6%	2780
6-layer CNN	2 / 2	MNIST	8	98.7%	5072

IV. Research Questions

RQ1: What is the impact on GANs performance when training across multiple clients?

RQ2: Can data privacy be preserved when GANs are collaboratively trained?

RQ3: Can we have a server that is "blind" to both the input and to the generated content?

RQ4: What is the performance of an FHE-enabled GAN?

V. Distributed GAN training (RQ1, RQ2)

A. Method

In order to measure the performance and security of a possible distributed learning scenario, we propose and experiment in which we train a state-of-the-art conditional GAN, the Pix2Pix model [20]. This model utilizes a U-Net style generator and a Markovian discriminator, termed PatchGAN in their paper.

In addition to the classical cGAN objective (Equation 1), Isola also used the classical loss function $L1$. The ultimate goal of the Pix2Pix model, with λ determining the influence of the L1 norm, can be expressed as follows:

$$\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G) \tag{2}$$

The authors have found that the best-performing model architecture (as seen in Figure 3), is with a discriminator patch size of 70x70.

The use case chosen for this experimental part is one of the use cases presented in the Pix2Pix paper [20], where the model is trained to generate aerial photo from a map image, using the data provided by the authors. This particular use case does not use sensitive data, but it was chosen so that it can be compared with the original results when trained with the same hyperparameters.

The scenario considered in this work is of three entities with the same amount of images. The data is split randomly into three batches, therefore it is independent and identically distributed (IID). Non-IID scenarios should also be analyzed in future work.

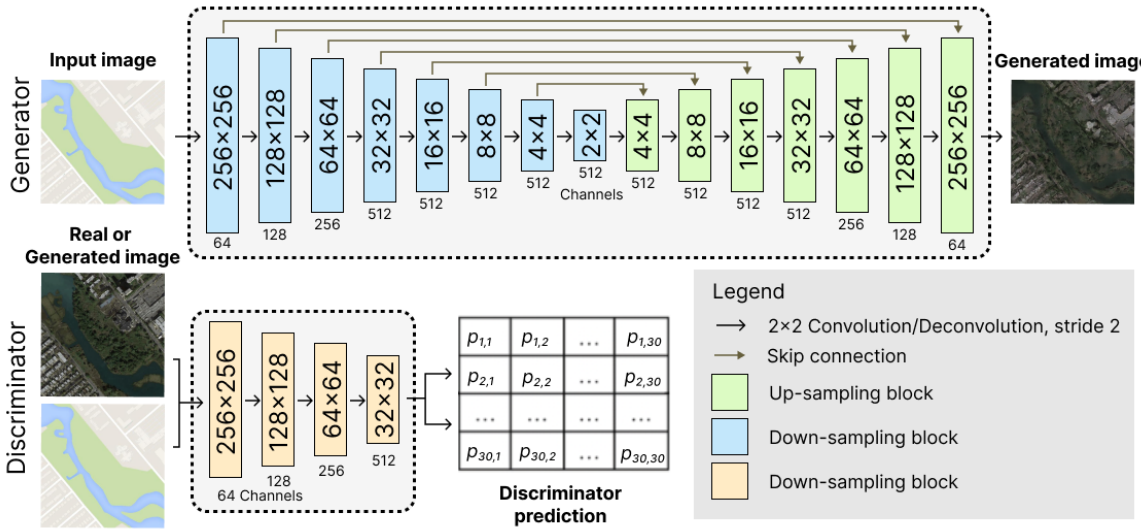


Fig. 3: The architecture of the Pix2Pix model.

We implemented three scenarios which we then compared. In the first experiment, we centralized the data in one site and trained the model locally, performing 200 training epochs over the whole data set, with a batch size of 1, as mentioned in the Pix2Pix paper. Clearly, this approach compromises all the data, or at least some part of it, if the central entity is one of the participating parties. However, this method was chosen to be tested in order to analyze the performance and compare it with further, more secure methods.

As one of the objectives of this work is to determine whether federated learning is worth implementing in a cross-silo scenario, a training run was performed on the data of a single client. The experiment has the same settings as the previous centralized case, only with a third of the data.

In the third experiment, we implemented a FL solution using the NVIDIA FLARE framework. The simulated local learning mode was used, in which the server and the three participating clients run on separate processes.

Lastly, a truly distributed experiment was performed using the Azure Machine Learning platform, creating a separate Virtual Machine (VM) for each previous process, the server, and the clients. The clients were created on VMs with NVIDIA Tesla M60 GPUs. The NVFLARE framework provides the communication and setup infrastructure needed to perform the distributed learning. We also applied differential privacy on the model updates by introducing Gaussian noise with $\sigma_0 = 0.1$.

In order to evaluate the performance of the generator models, we used the Fréchet inception distance (FID) [21], implemented in Python and maintained by Maximilian Seitzer and it is available on GitHub [22].

B. Results

The generator or discriminator loss are not objective metrics that can be used to analyze the performance of the model. However, it is worth comparing the model loss in the different settings we have described. Figure 4 shows the losses of these runs.

The local site training has the lowest loss, but, as will be seen further, it produces the worst results. This can be due to model overfitting, due to the limited data set. The model loss when adding differential privacy to federated learning closely follows the federated learning loss. This shows that the added noise does not cumber the generator in reaching its subjective goal. However, the generator takes a longer number of epochs to reach the same loss as the centralized model.

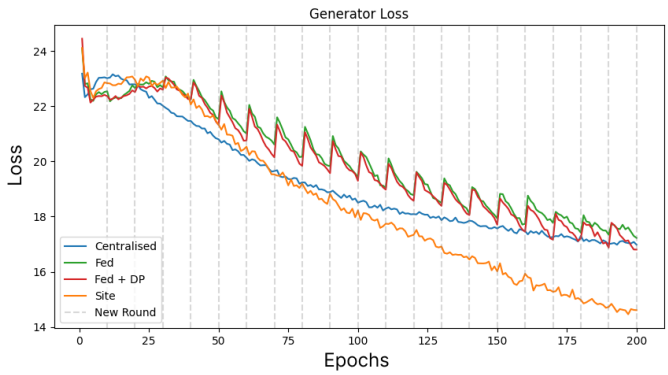


Fig. 4: Generator Loss Values Comparison

As one of the chosen metrics for this work is the FID score, the generator state was saved at epochs 50, 100, 150, and 200. In the case of federated learning runs, these correspond to rounds 5, 10, 15, and 20, respectively. These models were used to generate outputs for each of the evaluation images. These generated collections of images were then used to compute the FID scores seen in 6, comparing them to the ground truth of the evaluation data.

A problem with the computed FID scores is that models were evaluated over the whole data set, but such a metric is

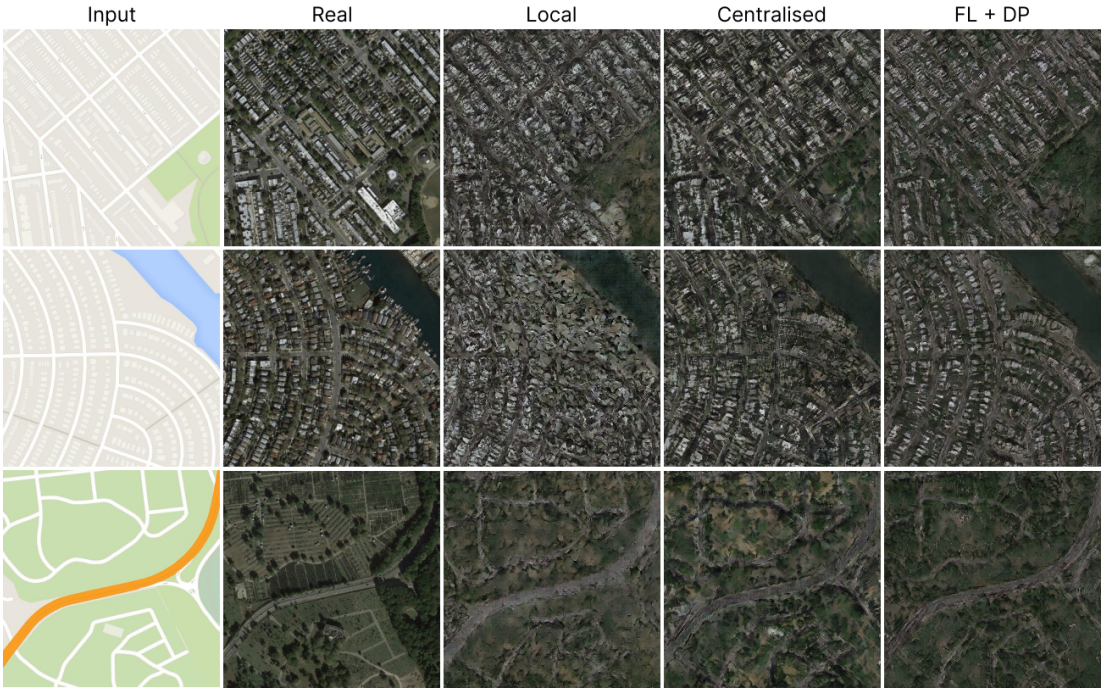


Fig. 5: Visual comparison of the different training methods

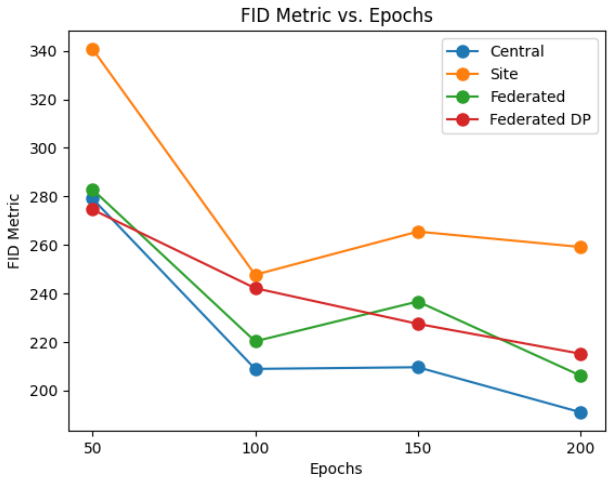


Fig. 6: FID scores over the whole data set (Lower is better)

not possible in a real-world scenario, as the evaluation images are private and stored at the participating sites. In order to evaluate the practicality of this approach, the final model of each training method was used to infer over the whole set of evaluation images of each site, storing the generated images locally at each site. The FID score was then computed for each of these generated sets. In Table II, it can be seen that the proportionality between the FID scores of the different training methods used is the same as in the centralized evaluation.

The other metric used to analyze the models is visual evaluation, as the goal is to produce images indistinguishable from reality by humans. Some of the generated images at

Site	Centralized	FL + DP	Local
North EU	204.01	229.49	268.49
West EU	202.99	233.44	272.72
US	207.09	230.64	274.81

TABLE II: FID scores over the local data after 200 epochs

the last model state are seen in Figure 5. The locally trained models are the farthest from the ground truth, as the model produces detailed but fuzzy images. The images trained in the Federated Learning setting with Differential Privacy are close in quality to the ones generated by the centralized model, albeit less detailed.

These visual results support the results of the FID scores comparison. Federated training produces better results than training locally, but there is a loss in quality compared to the centralized model.

VI. Private GAN Inference (RQ3, RQ4)

A. *Methods (RQ3)*

The majority of FHE models need to be quantized to meet FHE constraints, as FHE is limited to 16-bit computations. Quantization is the process of reducing an input of very large values (for example, real numbers) to a discrete set (for example, integers). Quantization reduces the precision of the computations done in a model. Usually, this is done by switching from floating-point representations (like FP32) to lower bit-width numbers, for instance, fixed-point (like INT8). In order to achieve the best accuracy, we are using Quantization Aware Training (QAT). The term ”aware” means that the model is being trained with the precision limitation

being imposed during training rather than after it. By making the model ”aware” of quantization, we allow it to adjust the weights to our needs.

Initially, we tried to implement the conditional Pix2Pix GAN above, by adding QAT, as we can see in Figure 7. The problem seemed to be that the discriminator learned way too quickly how to classify the images. We have tried to reduce the learning rate but to no avail. Unfortunately, it seemed like the model needed the precision in order to produce the images, which unfortunately was a deal-breaker for us. As we could not quantize the model successfully, we had to totally abandon all the FHE-related work done prior and start over with a new, smaller model.

(a) Input 1 (b) Output 1 (c) Input 2 (d) Output 2

Fig. 7: Generated images from the Pix2Pix Quantized model

That is why we have settled to go for a MNIST GAN, and started to implement it in Python using the library PyTorch, using [23] as a reference.

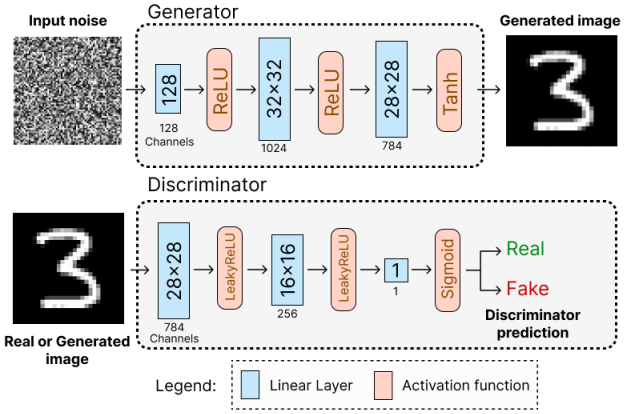


Fig. 8: Structure of MNIST GAN

We applied QAT using Brevitas on all the layers and activation functions.

To further help us with decreasing the FHE load, we use pruning. Pruning is a technique that removes certain parts of the network, such as weights, nodes, or even whole layers, therefore reducing the computational complexity of the model. As a pruned model usually has fewer parameters and connections, it will run more efficiently and occupy less memory, all while trying to maintain a comparable level of performance. Evidently, by cutting off information we can expect a drop in accuracy, but because we are removing only the most unimportant parts, we expect this drop to not be significant. For our experiments, we use L1 unstructured pruning.

We deploy the model on AWS with the help of ConcreteML scripts and we build a client to send requests. We will use powerful machines in order to see how it affects inference times.

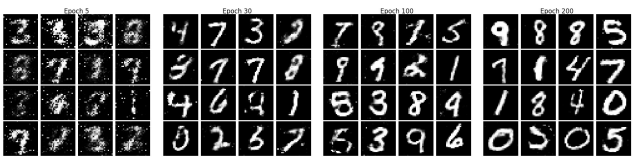
B. Results (RQ4)

1) Training: We have trained three different models, all using the same machine, locally. The device used was an Asus Zephyrus G14, CPU: Ryzen 7 4800HS, 16GB RAM, and a NVIDIA GeForce GTX 1650TI GPU with CUDA Enabled.

Generator Model Type	Training Time / 200 epochs (in seconds)
Regular Model	$3208s$
6-bit QAT Model	$7507s$
4-bit QAT Model	$7068s$

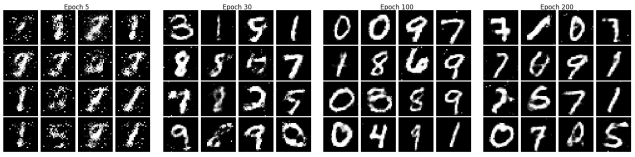
TABLE III: Training Times for Different Generators

The information in Table III suggests that on average, the Quantization Aware Training is around 2 times slower than the regular training. This is mainly because of Python's parallelization issues and Brevitas' lack of optimizations. Although these training times might be a problem for large models (for example, LLMs) where the training times are very large, we expect these issues to be optimized in the future.



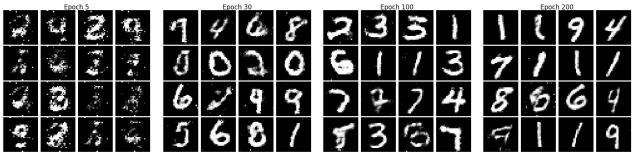
(a) Epoch 5 (b) Epoch 30 (c) Epoch 100 (d) Epoch 200

Fig. 9: Regular Generator



(a) Epoch 5 (b) Epoch 30 (c) Epoch 100 (d) Epoch 200

Fig. 10: 6-bit Quantization Aware Training Generator



(a) Epoch 5 (b) Epoch 30 (c) Epoch 100 (d) Epoch 200

Fig. 11: 4-bit Quantization Aware Training Generator

(a) Original (b) 20% pruned (c) 50% pruned (d) 80% pruned

Fig. 12: Pruning results

Model type	Pruned	FID
Original	$None$	57.1517
4-bit QAT	$None$	59.1477
4-bit QAT	20% pruning	60.7165
4-bit QAT	50% pruning	67.8442
4-bit QAT	70% pruning	108.1767
4-bit QAT	80% pruning	143.2431

TABLE IV: FID for Different Generators (Lower is better)

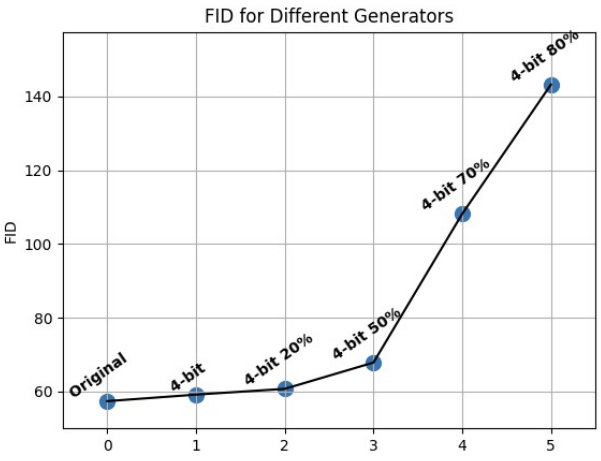


Fig. 13: Plot of FID (Lower is better)

We can see in Figures 9, 10, 11, 12, 13 and Table IV that the quality of photos is not much different whether we use an original, 6-bit, or 4-bit QAT model. Therefore, we choose the most efficient model, the 4-bit model. Next, we can see in Figure 13 that pruning affects the model minimally at 20% and 50% but drops off drastically at 70%. Therefore, we can consider that the 20% or 50% pruned models give the best balance between quality and performance.

2) Inference: We have run inference tests for all kinds of devices and models, in order to find find the best time. Therefore, we have considered software and hardware optimizations.

For the hardware, the following devices were used:

- Macbook Pro 16' M1 Pro, 16GB RAM, 10-core CPU
- AWS c5.large - 2 vCPUs, 4GB RAM
- AWS m6i.8xlarge - 32 vCPUs 3rd Gen Intel Xeon Scalable, 128GB RAM
- AWS m6i.metal - 128 vCPUs 3rd Gen Intel Xeon Scalable, 512GB RAM

It is worth noting that FHE is a CPU-intensive operation, therefore we have focused on having AWS instances that have as many vCPUs as possible. The GPUs are not used for inference, so we disregarded those when choosing the appropriate instances.

For the models used, we have the following options:

- Regular model - not used in the inference
- 6-bit QAT Model - tested locally
- 4-bit QAT Model - tested locally
- 4-bit QAT Model + 20% pruning - tested locally
- 4-bit QAT Model + 50% pruning - tested locally and on AWS

Server device	Generator Type	Inference Time (s)
Macbook Pro	6-bit QAT	$5721s$
Macbook Pro	4-bit QAT	$4823s$
Macbook Pro	4-bit QAT + 20% pruning	$4314s$
Macbook Pro	4-bit QAT + 50% pruning	$3780s$
AWS c5.large	4-bit QAT + 50% pruning	Insufficient memory
AWS m6i.8xlarge	4-bit QAT + 50% pruning	$2114s$
AWS m6i.metal	4-bit QAT + 50% pruning	$738s$

TABLE V: Inference Times

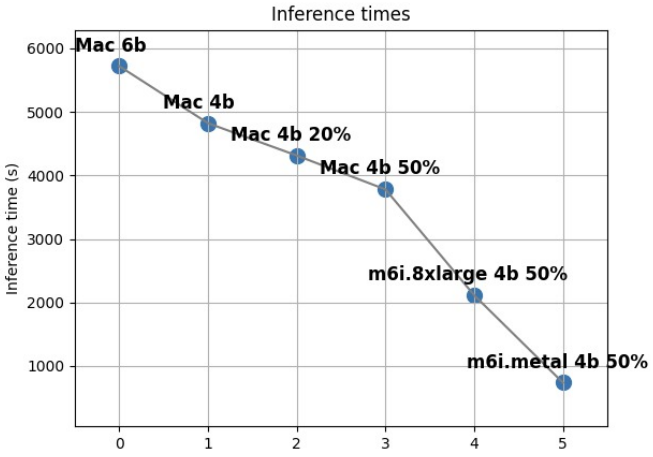


Fig. 14: Plot of Inference Times (Lower is better)

We can see in Figure 14 that our optimizations help the inference times tremendously. By switching from a 6-bit QAT to a 4-bit QAT we achieve a 1.3x speedup. Moreover, if we add the pruning, we achieve about a 1.6x speedup from what we initially had. Having a very powerful device to run the computations on proved to be one of the biggest factors in achieving a 7x speedup, which is the lowest inference time our experiments have found. We note that our first experiment ran an un-optimized 6-bit model on our local device running on lower-power mode, which is why our very first experiment timed at about 2 hours. However, we do not include this first measurement as we have re-run the experiments on the Macbook without the power limitations. These results show that having the right hardware and optimizations is having a massive impact on the performance of the inference, without losing much quality.

It might seem surprising that a machine with 128vCPUs and 512GB RAM needs a whole 12 minutes just to generate a 28x28 image, considering that inference in the clear takes about 0.01 seconds. However, this was expected from the beginning. These experiments show that FHE is around 5 orders of magnitude slower than computations in the clear, which is supported by state-of-the-art research on other types of Machine Learning models.

VII. CONCLUSIONS

Motivated to integrate privacy-preserving methods into high-performing generative models, in an era characterized by increasing data breaches, the aim of this work is to provide insight into how these models can be efficiently trained and inferred on while minimizing the exposure of private user information.

We begin our experiments by training the Pix2Pix model with different methods and comparing the obtained results. Answering RQ1, we show that training the model with FL provides better results than training only with the local data a client holds. However, the performance of the resulting model is still lower than the centralized approach, as we expected. Visual examples show that the model still produces good results even when applying Differential Privacy through Gaussian noise.

Combining DP with the security provided by the FLARE framework, we trained this model in the cloud, using Azure services. Obtaining qualitative results, evaluated both using FID scores and visual comparison, we conclude that the data used in training a GAN model with FL is not jeopardized (RQ2), while still obtaining performant models.

Regarding FHE inference for GANs, we have proven its possibility by building and deploying the model, however, with various limitations, such as the need for a simple model (RQ3). Overall, we have achieved a 7x speedup through various optimizations, but our lowest inference time was around 12 minutes, which proves that FHE as a technology is not yet practical (RQ4). There are various improvements that might help improve the FHE results: better optimizing the model, using alternative pruning methods, changing the structure of the generator, etc. Future work in FHE is mainly concerned with hardware optimizations. Our 12-minute inference time cannot be vastly improved with software or design improvements, even if we sacrifice more of the accuracy. Rather, in the following years, we can expect a 10.000x speedup from specialized FHE hardware processors, which will greatly improve our inference times.

All our findings point to one conclusion: are possible, but far away from being practical. Researchers expect FHE to be ready for real-life applications in about 5 years. However, we knew this from the beginning, and these results are not a discouragement: rather, they are a fascinating view into what the future might look like.

Privacy should not be a luxury, but a right. By employing these methods, we are proving that in the future, privacy within Generative AI could be **guaranteed by design**.

REFERENCES

[1] Cecily Mauran. Whoops, Samsung workers accidentally leaked trade secrets via ChatGPT. https://mashable.com/article/samsung-chatgpt-leak-details. Online; accessed 23 Apr 2023.

[2] Abeer Aljohani and Nawaf Alharbe. Generating synthetic images for healthcare with novel deep pix2pix gan. *Electronics*, 11(21), 2022.

[3] Jingzhang Sun, Yu Du, Chien-Ying Li, wu Hsin, Bang-Hung Yang, and Greta Mok. Pix2pix generative adversarial network for low dose myocardial perfusion spect denoising. *Quantitative Imaging in Medicine and Surgery*, 12, 01 2021.

[4] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[5] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[6] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[7] Luc Rocher, Julien Hendrickx, and Yves-Alexandre Montjoye. Estimating the success of re-identifications in incomplete datasets using generative models. *Nature Communications*, 10, 07 2019.

[8] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2023.

[9] Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

[10] Craig Gentry. Fully homomorphic encryption using ideal lattices. *Proceedings of the Annual ACM Symposium on Theory of Computing*, 9, 05 2009.

[11] Xun Wang, Tao Luo, and Jianfeng Li. A more efficient fully homomorphic encryption scheme based on gsw and dm schemes. *Security and Communication Networks*, 2018:1–14, 12 2018.

[12] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Tfhe: Fast fully homomorphic encryption over the torus. Cryptology ePrint Archive, Paper 2018/421, 2018. https://eprint.iacr.org/2018/421.

[13] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23*, pages 409–437. Springer, 2017.

[14] Arthur Meyre, Benoit Chevallier-Mames, Jordan Frery, Andrei Stoian, Roman Bredehoft, Luis Montero, and Celia Kherfallah. Concrete ML: a privacy-preserving machine learning library using fully homomorphic encryption for data scientists, 2022. https://github.com/zama-ai/concrete-ml.

[15] Alessandro, Giuseppe Franco, Ian Colbert, Andrei Stoian, nickfraser, Javier Duarte, Luis Montero, MichalMachura, Omar Peracha, SpontaneousDuck, Yaman Umuroglu, derpda, and vfdev. Xilinx/brevitas: Super resolution r0, May 2023.

[16] Chenyou Fan and Ping Liu. Federated generative adversarial learning. *arXiv preprint arXiv:2005.03793*, 2020.

[17] Ali Hatamizadeh, Hongxu Yin, Pavlo Molchanov, Andriy Myronenko, Wenqi Li, Prerna Dogra, Andrew Feng, Mona G. Flores, Jan Kautz, Daguang Xu, and Holger R. Roth. Do gradient inversion attacks make federated learning unsafe? *IEEE Transactions on Medical Imaging*, 2023.

[18] Filip Granqvist, Matt Seigel, Rogier van Dalen, Áine Cahill, Stephen Shum, and Matthias Paulik. Improving on-device speaker verification using federated learning with privacy. *arXiv preprint arXiv:2008.02651*, 2020.

[19] Andrei Stoian, Jordan Frery, Roman Bredehoft, Luis Montero, Celia Kherfallah, and Benoit Chevallier-Mames. Deep neural networks for encrypted inference with tfhe. Cryptology ePrint Archive, Paper 2023/257, 2023. https://eprint.iacr.org/2023/257.

[20] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2018.

[21] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *arXiv preprint arXiv:1706.08500*, 2018.

[22] Maximilian Seitzer. pytorch-fid: FID Score for PyTorch. https://github.com/mseitzer/pytorch-fid, August 2020. Version 0.3.0.

[23] PyTorch-Mnist-GAN. https://github.com/ccs96307/PyTorch-GAN-Mnist. Online; accessed 25 Feb 2023.

[24] Daisuke Matsuoka, Shiori Sugimoto, Yujin Nakagawa, Shintaro Kawahara, Fumiaki Araki, Yosuke Onoue, Masaaki Iiyama, and Koji Koyamada. Automatic detection of stationary fronts around japan using a deep convolutional neural network. *SOLA*, 15:154–159, 2019.