FEDERATEDINFERENCE: Federated Inferencing for Score-Based Diffusion Models

Luke Braithwaite
University of Cambridge
lb2027@cam.ac.uk

Matthew Hattrup
University of Cambridge
mh2236@cam.ac.uk

Abstract

Federated Learning has emerged as a means to train models over decentralized data. In the real world, data often cannot be centralized due to costs, privacy concerns, or regulation making centralized training in these cases impossible. Federated Learning is a potential solution for training in these contexts, whereby a central model learns a global distribution by aggregating training over isolated datasets. In this paper, we propose FEDERATEDINFERENCE framework, whereby models are pretrained on seperate datasets and their predictions are combined for collective inferencing. We adapt Score-Based diffusion models to a generative framework and experiment with different regimes where the agents are trained on homogeneous and heterogenous datasets. We show that in both the homogeneous and heterogeneous regimes, federated inferencing performs similarily. We also show that our FEDERATEDINFERENCE framework is more inconsistent than a centralized approach at producing correct digits, but when it does, the images are of a higher quality than a centralized approach. We suggest possible remedies in the discussion. In the future, FEDERATEDINFERENCE could be a useful tool for collective inferencing across models trained on isolated datasets.

1 Introduction

1.1 Federated Learning

Federated Learning (FL) has been applied to different tasks including mobile keyboard prediction [3], healthcare [18, 8] and autonomous driving [2, 27]. We adapt Bhagoji et al. [1]'s definition to define Federated Learning.

Definition 1 (Federated Learning)**.** In *federated learning* we have a set of agents $\{a_k\}_{k=1}^K$ which each has access to a dataset $\mathcal{D}_k$ that is not shared with the other agents or the centeral server. The server trains a model $f(\mathbf{w})$ where the parameter $\mathbf{w}$ obtained through

$$\mathbf{w}^{t+1} = \text{AGGREGATE}\left(\{\mathbf{w}_k^t\}_{k=1}^K\right), \tag{1}$$

where $\mathbf{w}_k^t$ is the weights of agent a_k trained on $\mathcal{D}_k$ and AGG is a (permutation invariant) aggregation function.

To give a more concrete example, consider the FEDERATEDAVERAGING [12] algorithm. This is the original federated learning approach and starts by modifying the SGD algorithm. During each training round, the model is trained for a fixed number of steps before the weights are sent back to the server to be aggregated during the federated averaging step. In this step the new model weights are calculated by averaging the agent's weights which are then sent back to the agents before the next round. For inferencing, the final federated weights are used.

Federated learning solves several key limitations traditional centeralised ML training processes.

Final Project Report. L46 Principles of Machine Learning Systems.

- In many applications such as internet of things or other embedded devices it may be infeasible to send all the data back to a centeralised server. FL requires only the model's weights to be sent to a central server instead of the entire dataset
- Regulations such as GDPR can prevent the transfer of user data across international borders. FL prevent this from being neccessary.
- FL preserves user privacy by preventing data sharing with third-parties, this is particularly important for sensitive data such as health or medical records as well as personal information like text messages and photos.

1.2 Federated Inferencing

We now develop the mathematical framework for federated inferencing. Consider again K agents. As in Federated Learning, each agent a_i has access to a dataset $\mathcal{D}_i$ which it does not share with the other agents. Unlike Federated Learning, we do not learn a model f on the central server. Instead we assume each agent a_i is pretrained on dataset $\mathcal{D}_i$. Given an input $\mathbf{x}_0$ we want to generate a final inference $\mathbf{x}_T$. At each timestep, we generate inference

$$\mathbf{x}_t = f_t(a_0(\mathcal{C}_{0,t}), a_1(\mathcal{C}_{1,t}), ...a_K(\mathcal{C}_{K,t})) \text{ for all } t \in (1, T], \tag{2}$$

where $\{f_t\}_t$ is a sequence of aggregation functions known to the server and $\{\mathcal{C}_{i,t}\}_{i,t}$ form a sequence of contexts supplied to each agent a_i. There are two important things to take away from this framework. One, agents require no additional training or weight synchronization as is typical of Federated Learning. Two, we make no assumptions on agent parameter or inputs spaces, so long as their outputs can be aggregated sensibly. In this paper, however, the parameter and input space is uniform across agents.

We motivate this framework with a simple use case. Consider two hospitals in seperate parts of a city. Each hospital has a model which takes in a patient's medical symptoms and outputs list of diagnoses and corrposing symptoms which a doctor can then use to inform their diagnosis. Because of privacy reasons, the hospitals can't share their medical records with each other but they are able to share anonmyized patient symptoms to each others' models. In this example, the hospitals are agents a_0, a_1 and there are two timesteps $t \in [0, 1]$. The contexts are $\mathcal{C}_{i=0,t=1} = \mathcal{C}_{i=1,t=1} = \mathbf{x}_0$, where $\mathbf{x}_0$ is the patient's symptoms and $\mathbf{x}_{T=1}$ is the concatentation of each agent's prediction. One day, a patient comes in with an illness which is fairly uncommon at this hospital but has been seen before at the other. The second hospital's model recognizes the patient's symptoms and offers a correct diagnoisis.

1.3 Score-Based Generation through SDEs

Diffusion models are a class of state-of-the-art generative models for images [16, 17], audio [9], graphs [14], and molecular synthesis [7] and more. Diffusion models are a general framework encompassing *Denoising probabilistic diffusion modelling* (DDPM) [6], *Denoising probabilistic implicit modelling* (DDIM) [22], *Score-matching with Langevin dynamics* (SMLD) [23], and *Score-based generative modelling through stochastic differential equations* (SBGM-SDE) [24] and others. In each model, a diffusion process takes data from a target distrubition and gradually adds noise to it until it appears as though the point was sampled from a Gaussian distribution. We then learn a reverse diffusion process, creating a generative model of the target distribution given Gaussian noise. The diffusion process itself is very general and is defined differently in each framework.

In both SMLD and SBGM-SDE, a model learns to estimate the score (i.e the gradient of the log probability density function of the target distribution). We focus our discussion on SBGM-SDE, which constructs a continous diffusion process over the interval $t \in [0, T]$, defined by the SDE

$$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\,\mathrm{d}t + g(t)\,\mathrm{d}\mathbf{w}, \tag{3}$$

where $\mathbf{w}$ follows a standard Weiner process [25], $\mathbf{f}(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d$ is the drift coefficient, and $g(t) : \mathbb{R} \to \mathbb{R}$ is the diffusion coefficient. The drift coefficient governs the expected flow of the SDE and the diffusion coefficient governs the variance (i.e the randomness) of the SDE. Running the diffusion process backwards gives another diffusion process from T to 0 defined by the SDE

$$\mathrm{d}\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})]\,\mathrm{d}t + g(t)\,\mathrm{d}\tilde{\mathbf{w}}, \tag{4}$$

where $\tilde{\mathbf{w}}$ is standard Weiner process in reverse time, $p_t(\mathbf{x})$ is the probability density of $\mathbf{x}(t)$, and $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ is the score function to be learned by a neural network. In practice, the reverse-time

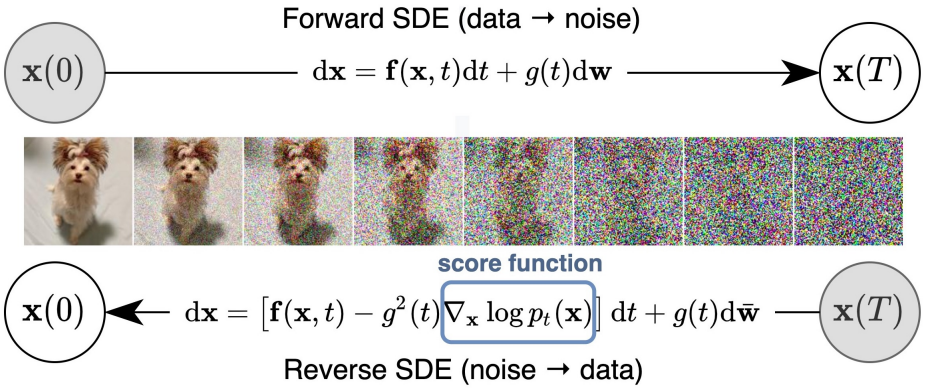

Figure 1: **Demonstration of a score-based diffusion model on image data.** The forward SDE models the continuous addition of noise to the image. Reversing the SDE yields a score-based diffusion model where the score $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ is learnt by a neural network. Source: [24, Fig. 1]

SDE can be solved using any off the shelf ODE solver. Figure 1 demonstrates what this diffusion process would look like for images.

2 Related Work

Like FL, federated inferencing provides a way to process models that are trained on isolated datasets. Unlike FL, federated inferencing is used to perform inferencing across pre-trained agents. Instead of aggregating weights, we aggregate the model outputs in a way analagous to ensemble techniques such as bagging. There are several possible generative diffusion framework that may work with federated inferencing. The DDPM and DDIM are popular choices for diffusion models. In both cases, the forward diffusion process is a discrete sequence of random variabels defined by a conditional Gaussians distrubitions. That is,

$$p(\mathbf{X}_t \mid \mathbf{X}_{t-1}) = \mathcal{N}(\mathbf{X}_t; \sqrt{\alpha_t}\mathbf{X}_{t-1}, \beta_t \mathbf{I}), \tag{5}$$

where $\{\alpha_t\}_t$ and $\{\beta_t\}_t$ are predefined scaling and variance schedules such that $\alpha_t = 1 - \beta_t$ [6]. At each time-step in the reverse diffusion process, each agent takes a step according to a learned Gaussian. In expectation, each agent will take a step bringing it closer to the learned mean (highest likelihood). If each agent learns a similar reverse diffusion process, then we would expect the steps taken by each agent to constructively add to one another. Therefore, DDPM and DDIM should both be possible to use for federated inferencing, but we instead went with SBGM-SDE [24].

3 Federated Inferencing with Score-based Diffusion Models

Consider a set of K sample distributions p_i sampled from a global distribution Γ. There are K agents $a_i(\mathbf{x}, t)$ which estimate the score function $\nabla_{\mathbf{x}} \log p_{i,t}(\mathbf{x})$. We can state the following key result.

Theorem 1. *If $p_i \approx p_j$ for all $i, j \in [1, K]$ and $\Gamma \approx \frac{1}{K}(\sum_i p_i)$, then,*

$$\nabla_{\mathbf{x}} \log \Gamma_t(\mathbf{x}) \approx \frac{1}{K} \sum_i \frac{\nabla_{\mathbf{x}} p_{i,t}(\mathbf{x})}{p_{i,t}(\mathbf{x})} \tag{6}$$

Proof. See Appendix A. □

It follows that we can average the score-functions for p_i in a reverse diffusion process (4) to recover a generative process for Γ. Using the FEDERATEDINFERENCE framework in Section 1.2, we have the following steps to generate a sample $\mathbf{x}$:

1. Pretrain an ensemble of score-based models $\{a_i\}_i, i \in [1, K]$.

2. Define $\mathcal{C}_{i,t-\mathrm{d}t} \coloneqq (\mathbf{x}_t, t)$.
3. Define $x_{t-\mathrm{d}t} = f_{t-\mathrm{d}t}(\mathbf{x}_t) \coloneqq \mathbf{x}_t + \frac{1}{K}\sum_i \mathrm{d}\mathbf{x}_i$, where $\mathrm{d}\mathbf{x}$ is defined in Eq. 4 and K is the number of agents.

We can then $\mathbf{x}_0$ given $\mathbf{x}_T$ using an off the shelf ODE solver. Note the slight abuse of notation, the language of Section 1.2 specifies a discrete forward-time process, whereas Eq. (4) is a continous backward-time process. The direction of time isn't an issue and can be fixed by reindexing the contexts $\{\mathcal{C}_{i,t}\}_{i,t}$ and aggregation functions $\{f_t\}_t$. In practice, ODE-solver operate with discrete time anyway. Depending on the ODE solver, $\mathrm{d}t$ can be specified by a pre-define sequence of intervals or computed by the ODE solver like in Runge-Kutta methods.

4 Experimental Results

4.1 Experimental Setup

For this experiment we trained an SBGM-SDE diffusion model on the MNIST dataset using the diffusers library [15] by huggingface. The score model is implemeneted using the diffuser library's `UNet2DConditionModel` which is a modified time-conditioned U-Net [19] with a FreeU [21] mechanism[1]. We trained the model for 10 epochs using the AdamW optimiser [11] and a cosine learning rate schedululer with warm ups.

In order to explore the effect of heteogeity in the dataset we trained the diffusion model under the following three regimes.

CONTROL. In this regime on a single is agent is trained on the entirety of the MNIST dataset.

FEDERATEDINFERENCEHOMOGENEOUS. In this regime we train 10 agent where each agent's training samples composed of 10 % of each digit.

FEDEREATEDINFERENCEHETEROGENEOUS. In this regime we train 10 agent where each agent's training samples composed of 50 % of a single digit with the remaining 50 % being an equal mix of the remaining 9 digit classes. This regime was designed to explore the effect of statistical hetreogeneity in the distributions.

To compare the performance of each diffusion model we first generated 10 samples from each digit class and then calculated the average *Inception Score* (IS) [20] over each sample as well as the *Fréchet Inception Distance* (FID) [5] over the entire samples compared to ground truth images. The inception score measures how realistic the generated images are. Formally it is defined as,

$$IS = \exp(\mathbb{E}_x \,\mathrm{KL}(p(y|x) \parallel p(y))) \tag{7}$$

where $\mathrm{KL}(p(y|x) \parallel p(y))$ is the KL divergence between the conditional distribution $p(y|x)$ and the marginal distibution $p(y)$. Both of these distributions are calculated using features extracted from the images by the feature extractor. The FID is another way to assess the quality of the generated images, but instead the features extracted from the generated samples are compared to those extracted from the ground truth images. Formally we have,

$$FID = \|\boldsymbol{\mu} - \boldsymbol{\mu}_w\|^2 + \mathrm{Tr}\Big(\boldsymbol{\Sigma} + \boldsymbol{\Sigma}_w - 2(\boldsymbol{\Sigma}\boldsymbol{\Sigma}_w)^{\frac{1}{2}}\Big), \tag{8}$$

where $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is a multivariate gaussian distribution estimated on the features extacted from the ground truth images and $\mathcal{N}(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)$ is a multivariate gaussian calculated on the features extracted from the generated samples.

4.2 Quantatative Results

Table 1 shows the performance of FEDERATEDINFERENCING on both of the two training domains. Clearly FEDERATEDINFERENCING substantially underperforms compared to training a single model on both of the quantative performance metrics. We calculate both metrics using the implementations provided by *TorchMetrics* [13], each of these implementations required a custom feature extractor to

[1]We specifiy the exact model architecture in Appendix B

Table 1: Quantative results of our experiments. We report the average inception score (IS) over the generated sample along with the standard deviation and the Fréchet inception distance (FID) for each model. Clearly the control model significantly outperforms both FEDERATEDINFERENCING settings. We discuss why this is later on.

Model	IS ↑	FID ↓
CONTROL	**5.9459 ± 1.1504**	**1.6274**
FEDERATEDINFERENCEHOMOGENEOUS	3.6795 ± 1.0987	31.7551
FEDERATEDINFERENCEHETEROGENEOUS	3.1651 ± 0.8203	37.1922

geneerate the features needed to calculate the metric values. Our feature extractor was a ResNet18 [4] model with pretrained weights that can be found at `https://github.com/sundyCoder/IS_MS_SS` In Section 4.3 we quantatively evaluate the samples generated from each of our training regimes to explore the possible reasons for this poor performance.

4.3 Qualatative Results

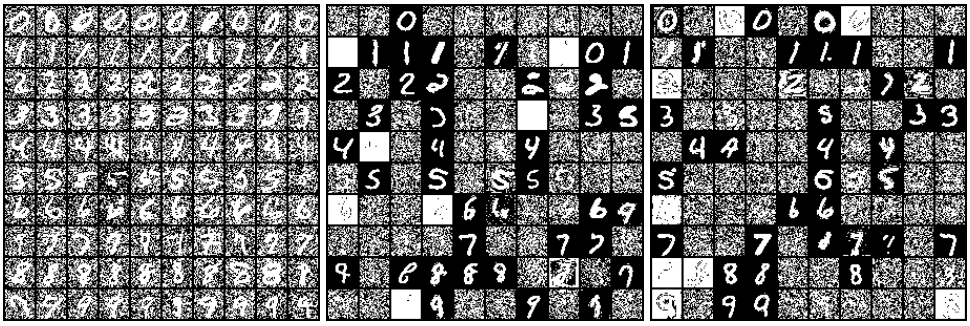

Figure 2: **Generated samples for each of the three training regimes.** For each model we generate 10 samples for each digit. **(LEFT)** CONTROL. **(CENTRE)** FEDERATEDINFERENCEHOMOGENEOUS. **(RIGHT)** FEDERATEDINFERENCEHETEROGENEOUS. It is now very clear why our method performs so poorly, most of the genrated samples appear to be gaussian noise.

Figure 2 shows the samples generated under each regime used for the quantative analysis. We can now clearly see the reason for the poor quantative performance achieved in Table 1, over 50 % of the samples generated by the FEDERATEDINFERENCE regimes were noise. But when the regimes does not generate the noise the samples generated are substantially less noisy than the control experiment.

Given that FEDERATEDINFERENCE can generate clean samples, the very noisy cases are most likely due to an issue with reverse diffusion process. It appears that the issue is the ODE solver gets stuck and it unable to remove any more noise from the image. We explore this idea in more detail in Section 4.4 by analysing the reverse SDE in more detail to understand why the solver may get stuck.

4.4 Analysing the Reverse Diffusion Process

We hypothesised the reason the homogeneous and heterogeneous regimes frequently generated noisy images was that they remain in a area of high noise in the beginning of the reverse diffusion process. This could be because the score estimates produced by each agent can cancel each other out when averaged. This would also explain why the control model never produces noise as no opposing agent score estimates can cancel it out. Later on in reverse diffusion process, estimates of the score are too sensitive for the amount of noise in the image and we remain trapped in the noisy valley. To test this hypothesis we evaluated the score function of each model in each regime, for every conditional label 0-9, for every time in $\{1.0, 0.5, 0.1, 0.01\}$, for 10 samples of random noise and 10 random samples from the MNIST training dataset which matched the conditional label.

For each sample in a batch (associated a specific regime, time, and conditional label, and whether or not the inputs were noise or images of digits which matched the conditional label) the score-function

was the average of score functions produced by each agent (in the control there is only 1 agent). We also measured the cosine-similarity between the average score-function and the score-function of each agent. We then average over each agent to get the average consine-similarity for that sample. The average magnitude and consine similarity per sample were averaged over each batch of 10 samples.

Formally, for each regime r, conditional label $l \in [0, 9]$, time $t \in \{1.0, 0.5, 0.1, 0.01\}$, $n \in \{0, 1\}$ (which indicates if the batch is of noise or of images corrosponding to the conditional label), agent a_i, and sample s_j. We have the following,

$$\mathbf{v}^r_{l,t,n,i,j} = a^r_i(s_{l,t,n,j}, t, l), \tag{9}$$

$$\bar{\mathbf{v}}^r_{l,t,n,j} = \frac{1}{K}\sum_i \mathbf{v}^r_{l,t,n,i,j}, \tag{10}$$

$$\bar{m}^r_{l,t,n} = \frac{1}{S}\sum_j \left\|\bar{\mathbf{v}}^r_{l,t,n,j}\right\|, \tag{11}$$

$$c^r_{l,t,n,j} = \frac{1}{K}\sum_i \frac{\mathbf{v}^r_{l,t,n,i,j} \cdot \bar{\mathbf{v}}^r_{l,t,n,j}}{\left\|\mathbf{v}^r_{l,t,n,i,j}\right\| \cdot \left\|\bar{\mathbf{v}}^r_{l,t,n,j}\right\|}, \tag{12}$$

$$\bar{c}^r_{l,t,n} = \frac{1}{S}\sum_j c^r_{l,t,n,j}, \tag{13}$$

where $S = 10$ is the number of samples per batch, $\bar{m}^r_{l,t,n}$ is the average magnitude per batch, and $\bar{c}^r_{l,t,n}$ the average cosine similarity per batch. The average magnitude and consine similarity per batch are shown in Tables 3 and 4.

5 Discussion

Perhaps most suprising are the qualtative results in Section 4.3. The control regime consistently produced noisey, yet recongnizable images of digits. Yet the homogeneous and heterogeneous regimes appear to produce very noisy images, almost entirely white images, or very high quality images of digits with very little in between. We hypothesized that this was because in areas of high noise, the predictions of each model may cancel out. Later on in the reverse diffusion process, the agents are too sensitive to noise to accurately predict a score estimate, so the agents never escape the noisy area. However this is not completely true, and on closer inspection the very noisy images they are not a just gaussian noise. Many of them contain shapes and faint outlines which form the beginnings of digits.

Consider the output space of 28x28 images. Some regions have high probabilies of being a particular digit, while much of the space is noise. Tables 3 and 4 show that in areas of high noise, the direction of each agent's score function are very similar (as measured by cosine similarity), and so they do not cancel out as predicited. Conversely, in areas resembling digits the score function's tend to be very dissimilar. The dissimilarity is more pronounced in the hetergeneous model. Also note that as time approaches 0, the magnitude and variance of score estimates increases. In other words, near the end of the reverse process, it becomes more urgent to each agent to leave the region so they take bigger steps, but there is also more disagreement on which direction to take.

We theorize that the agents initially work well to step out of noisy regions but then get stuck as they approach regions of digits. Each agent has its own idea of what a particular digit should look like, so as each agent approaches a region of digits their score estimates become more sensitive and being to diverge. As a result the regime stalls as it gets pulled in multiple directions at once. The contributions of each agent's score function cancel out leading to a non-productive gradient step and the model remains trapped between regions near the digit.

Incredibly, when the model does produce recognizable digits, they are much more clear than the digits produced by the control model. We speculuate that when the homogeneous and heterogeneous regimes happen to reach a region of a digit, some of the models will try to pull the regime away to an area where they think there is a higher probability of finding the digit. At the same time, the rest of the agents want to stay where they are. This tug-of-war forces the agents compromise, ending up in a region with only the most essential aspects of that digit, culling the background noise. We believe

this tug-of-war is also what causes the regime to stall in the first place. If we can delay when this tug-of-war takes place, the homogeneous and heterogeneous could consistently produce images of higher quality than the centralized regime. In the future, it is worth considering a more sophisitcated aggregation function. In the beginning of the reverse diffusion process, it may only need to sample from a single model to get out of a region with noise. As it closes in on the region of interest it includes more agents, averaging out the biases of each agent and ending up a higher quality output. This is analagous to the theory underpinning ensemble methods such as bagging [10, 26].

Finally, it's worth mentioning that the heterogeneous regime performs similarly to the homogenous regime both quanitatively, and qualitatively as seen in Table 1 and Fig. 2. This is despite each agent's distribution being widely different in the heterogeneous regime, suggesting that in practice we can relax the condtions of Theorem 1.

6 Conclusion and Future Directions

In this paper we present the FEDERATEDINFERENCE framework which we apply to score-based diffusion models. FEDERATEDINFERENCE appears to work with both homogeneous and heterogenous distributions. In both cases, the FEDERATEDINFERENCE regimes produced digit of significantly higher quality than in the centralized setting. However, the regimes frequently produces images that were unrecognizable. We believe adopting a slightly more sophisticated aggregration function may fix the instability in digit quality. It may be worth considering a hybrid approach, where the beginning of the diffusion process is run using only a single or very few agents and later in the diffusion process more agents are brought in to correct for biases in the final output image.

6.1 Future Directions

There are several potential future directions:

- Perform more experiments to gather more empirical evidence to support the weakening of the assumptions of Theorem 1.
- Explore approaches to fix the numerical instability in the inferencing process by developing more sophisticateds aggregating functions.
- The FEDERATEDINFERENCE method-agnostic framework, so future work could adapt this framework to other model architectures and methods.

Author Contributions

MH: Initial concept, implementation, formal analysis (Theorem 1 and proof), analysis of reverse SDE process, writeup (§1.2, §1.3, §3, §4.4, §5, §6)

LB: Implementation, quantative and qualtative analysis, writeup (§1.1, §2, §4.1, §4.2, §4.3, §5, §6.1)

References

[1] Arjun Nitin Bhagoji et al. *Analyzing Federated Learning through an Adversarial Lens*. 2019. arXiv: 1811.12470 [cs.LG].

[2] Shenghong Dai et al. 'Online Federated Learning based Object Detection across Autonomous Vehicles in a Virtual World'. In: *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*. 2023, pp. 919–920.

[3] Andrew Hard et al. *Federated Learning for Mobile Keyboard Prediction*. 2018. arXiv: 1811.03604 [cs.CL].

[4] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[5] Martin Heusel et al. 'GANs trained by a two time-scale update rule converge to a local nash equilibrium'. In: *Advances in neural information processing systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017.

[6] Jonathan Ho, Ajay Jain and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: 2006.11239 [cs.LG].

[7] Emiel Hoogeboom et al. *Equivariant Diffusion for Molecule Generation in 3D*. 2022. arXiv: 2203.17003 [cs.LG].

[8] Madhura Joshi, Ankit Pal and Malaikannan Sankarasubbu. 'Federated Learning for Healthcare Domain - Pipeline, Applications and Challenges'. In: *ACM Trans. Comput. Healthcare* 3.4 (2022).

[9] Zhifeng Kong et al. *DiffWave: A Versatile Diffusion Model for Audio Synthesis*. 2021. arXiv: 2009.09761 [eess.AS].

[10] Brieman Leo. 'Bagging predictors'. In: *Machine Learning* 24 (2004), pp. 123–140.

[11] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. 2019. arXiv: 1711.05101 [cs.LG].

[12] H. Brendan McMahan et al. *Communication-Efficient Learning of Deep Networks from Decentralized Data*. 2016. arXiv: 1602.05629 [cs.LG].

[13] Nicki Skafte Detlefsen et al. *TorchMetrics - Measuring Reproducibility in PyTorch*. 2022.

[14] Chenhao Niu et al. *Permutation Invariant Graph Generation via Score-Based Generative Modeling*. 2020. arXiv: 2003.00638 [cs.LG].

[15] Patrick von Platen et al. *Diffusers: State-of-the-art diffusion models*. https://github.com/huggingface/diffusers. 2022.

[16] Aditya Ramesh et al. *Hierarchical Text-Conditional Image Generation with CLIP Latents*. 2022. arXiv: 2204.06125 [cs.CV].

[17] Aditya Ramesh et al. *Zero-Shot Text-to-Image Generation*. 2021. arXiv: 2102.12092 [cs.CV].

[18] Nicola Rieke et al. 'The future of digital health with federated learning'. In: *npj Digital Medicine* 3.1 (2020), p. 119.

[19] Olaf Ronneberger, Philipp Fischer and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV].

[20] Tim Salimans et al. 'Improved Techniques for Training GANs'. In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. Curran Associates, Inc., 2016.

[21] Chenyang Si et al. *FreeU: Free Lunch in Diffusion U-Net*. 2023. arXiv: 2309.11497 [cs.CV].

[22] Jiaming Song, Chenlin Meng and Stefano Ermon. *Denoising Diffusion Implicit Models*. 2022. arXiv: 2010.02502 [cs.LG].

[23] Yang Song and Stefano Ermon. *Generative Modeling by Estimating Gradients of the Data Distribution*. 2020. arXiv: 1907.05600 [cs.LG].

[24] Yang Song et al. *Score-Based Generative Modeling through Stochastic Differential Equations*. 2021. arXiv: 2011.13456 [cs.LG].

[25] Norbert Wiener. 'Differential-Space'. In: *Journal of Mathematics and Physics* 2.1-4 (1923), pp. 131–174. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sapm192321131.

[26] Danny Wood et al. 'A Unified Theory of Diversity in Ensemble Learning'. In: *Journal of Machine Learning Research* 24.359 (2023), pp. 1–49.

[27] Hongyi Zhang, Jan Bosch and Helena Holmström Olsson. 'Real-time End-to-End Federated Learning: An Automotive Case Study'. In: *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*. 2021, pp. 459–468.

A Proof Theorem 1

Proof. The proof is simply

$$
\begin{aligned}
\nabla_{\mathbf{x}} \log \Gamma_t(\mathbf{x}) &\approx \nabla_{\mathbf{x}} \log \left(\frac{1}{K} \sum_i p_{i,t}(\mathbf{x}) \right) && \Gamma \approx \frac{1}{K} \sum_i p_{i,t}(\mathbf{x}) \\
&= \nabla_{\mathbf{x}} \log \left(\sum_i p_{i,t}(\mathbf{x}) \right) && \text{log rules and derivative of a constant} \\
&= \frac{1}{\sum_i p_{i,t}(\mathbf{x})} \sum_i \nabla_{\mathbf{x}} p_{i,t}(\mathbf{x}) && \text{chain rule} \\
&\approx \frac{1}{K} \sum_i \frac{\nabla_{\mathbf{x}} p_{i,t}(\mathbf{x})}{p_{i,t}(\mathbf{x})} && p_{i,t} \approx p_{j,t} \forall i, j \in [1, K]
\end{aligned}
$$

□

B `UNet2DConditionModel` Architecture of the Diffusion Model

The complete model architecture used to initialised diffuser's `UNet2DConditionModel` is given in Table 2.

Table 2: Initialisation parameters for the `UNet2DConditionModel` used in this paper

Parameter	Value
`sample_size`	(28, 28)
`in_channels`	1
`out_channels`	1
`layers_per_block`	2
`block_out_channels`	(32, 480, 576, 576)
`cross_attention_dim`	10
`down_block_types`	`["DownBlock2D", "DownBlock2D", "CrossAttnDownBlock2D", "DownBlock2D"]`
`up_block_types`	`["UpBlock2D", "UpBlock2D", "CrossAttnUpBlock2D", "UpBlock2D"]`

C Reverse Diffusion Processes Across FEDERATEDINFERENCE Regimes

Table 3: The average magnitude (Eq. 11) and cosine similairy (Eq. 13) per batch of 10 samples. Digits 0-4.

	Model	Digit									
		0		1		2		3		4	
		Magnitude	Cosine	Magnitude	Cosine	Magnitude	Cosine	Magnitude	Cosine	Magnitude	Cosine
$t = 1$											
	Control	2.86	1.00	2.88	1.00	2.81	1.00	2.84	1.00	2.82	1.00
Noise	Homogeneous	2.93 ± 0.05	1.00	2.91 ± 0.04	1.00	2.84 ± 0.05	1.00	2.98 ± 0.07	1.00	2.97 ± 0.04	1.00
	Heterogenous	2.97 ± 0.04	1.00	2.96 ± 0.06	1.00	2.97 ± 0.03	1.00	3.05 ± 0.04	1.00	2.92 ± 0.04	1.00
	Control	9.96	1.00	11.29	1.00	9.00	1.00	10.77	1.00	10.84	1.00
Label	Homogeneous	4.36 ± 3.05	0.30 ± 0.25	4.80 ± 3.72	0.36 ± 0.31	4.72 ± 3.46	0.34 ± 0.28	4.11 ± 2.25	0.30 ± 0.24	4.53 ± 3.68	0.33 ± 0.28
	Heterogenous	3.18 ± 2.51	0.15 ± 0.22	3.61 ± 2.95	0.15 ± 0.35	3.24 ± 2.55	0.12 ± 0.26	3.13 ± 2.35	0.13 ± 0.23	3.11 ± 2.35	0.16 ± 0.30
$t = 0.5$											
	Control	14.06	1.00	13.79	1.00	14.51	1.00	14.43	1.00	13.42	1.00
Noise	Homogeneous	14.54 ± 0.15	1.00	14.34 ± 0.21	1.00	14.59 ± 0.21	1.00	14.91 ± 0.14	1.00	14.22 ± 0.21	1.00
	Heterogenous	14.30 ± 0.20	1.00	14.56 ± 0.20	1.00	14.61 ± 0.17	1.00	14.31 ± 0.24	1.00	13.99 ± 0.21	1.00
	Control	10.89	1.00	13.04	1.00	9.44	1.00	11.29	1.00	11.44	1.00
Label	Homogeneous	4.36 ± 3.05	0.30 ± 0.25	4.80 ± 3.72	0.36 ± 0.31	4.72 ± 3.46	0.34 ± 0.28	4.11 ± 2.25	0.30 ± 0.24	4.53 ± 3.68	0.33 ± 0.28
	Heterogenous	3.18 ± 2.51	0.15 ± 0.22	3.61 ± 2.95	0.15 ± 0.35	3.24 ± 2.55	0.12 ± 0.26	3.13 ± 2.35	0.13 ± 0.23	3.11 ± 2.35	0.16 ± 0.30
$t = 0.1$											
	Control	81.16	1.00	81.69	1.00	78.51	1.00	84.99	1.00	82.70	1.00
Noise	Homogeneous	58.77 ± 2.42	1.00	59.94 ± 3.09	1.00	59.94 ± 3.36	1.00	59.12 ± 3.45	1.00	61.80 ± 2.55	1.00
	Heterogenous	59.18 ± 2.72	1.00	61.47 ± 2.57	1.00	59.60 ± 3.21	1.00	61.39 ± 3.60	1.00	62.54 ± 3.25	1.00
	Control	12.25	1.00	16.34	1.00	9.99	1.00	11.67	1.00	11.95	1.00
Label	Homogeneous	4.36 ± 3.05	0.30 ± 0.25	4.80 ± 3.72	0.36 ± 0.31	4.72 ± 3.46	0.34 ± 0.28	4.11 ± 2.25	0.30 ± 0.24	4.53 ± 3.68	0.33 ± 0.28
	Heterogenous	3.18 ± 2.51	0.15 ± 0.22	3.61 ± 2.95	0.15 ± 0.35	3.24 ± 2.55	0.12 ± 0.26	3.13 ± 2.35	0.13 ± 0.23	3.11 ± 2.35	0.16 ± 0.30
$t = 0.01$											
	Control	188.27	1.00	187.31	1.00	187.26	1.00	188.62	1.00	185.76	1.00
Noise	Homogeneous	73.34 ± 6.75	0.99	74.36 ± 5.47	0.99	73.51 ± 6.37	0.99	74.53 ± 5.32	0.99	73.97 ± 5.92	0.99
	Heterogenous	71.56 ± 4.78	0.99	73.44 ± 4.79	0.99	73.33 ± 4.08	0.99	72.97 ± 5.18	0.99	73.50 ± 4.01	0.99
	Control	12.45	1.00	16.81	1.00	10.06	1.00	11.69	1.00	11.99	1.00
Label	Homogeneous	4.36 ± 3.05	0.30 ± 0.25	4.80 ± 3.72	0.36 ± 0.31	4.72 ± 3.46	0.34 ± 0.28	4.11 ± 2.25	0.30 ± 0.24	4.53 ± 3.68	0.33 ± 0.28
	Heterogenous	3.18 ± 2.51	0.15 ± 0.22	3.61 ± 2.95	0.15 ± 0.35	3.24 ± 2.55	0.12 ± 0.26	3.13 ± 2.35	0.13 ± 0.23	3.11 ± 2.35	0.16 ± 0.30

Table 4: The average magnitude (Eq. 11) and cosine similairy (Eq. 13) per batch of 10 samples. Digits 5-9.

	Model	Digit									
		5		6		7		8		9	
		Magnitude	Cosine	Magnitude	Cosine	Magnitude	Cosine	Magnitude	Cosine	Magnitude	Cosine
$t = 1$											
	Control	2.87	1.00	2.93	1.00	2.85	1.00	2.87	1.00	2.85	1.00
Noise	Homogeneous	2.89 ± 0.05	1.00	3.10 ± 0.03	1.00	2.87 ± 0.04	1.00	3.00 ± 0.03	1.00	2.89 ± 0.05	1.00
	Heterogenous	2.88 ± 0.05	1.00	2.88 ± 0.04	1.00	2.91 ± 0.04	1.00	2.88 ± 0.03	1.00	3.07 ± 0.07	1.00
	Control	10.67	1.00	12.18	1.00	12.17	1.00	9.60	1.00	11.08	1.00
Label	Homogeneous	4.37 ± 2.93	0.29 ± 0.26	4.33 ± 2.93	0.30 ± 0.25	4.44 ± 3.13	0.32 ± 0.27	4.38 ± 3.07	0.29 ± 0.26	4.44 ± 3.15	0.31 ± 0.27
	Heterogenous	3.15 ± 2.37	0.12 ± 0.26	3.20 ± 2.44	0.14 ± 0.34	3.26 ± 2.62	0.13 ± 0.27	3.12 ± 2.36	0.14 ± 0.26	3.41 ± 2.82	0.16 ± 0.24
$t = 0.5$											
	Control	14.40	1.00	15.18	1.00	14.32	1.00	14.97	1.00	14.34	1.00
Noise	Homogeneous	14.40 ± 0.16	1.00	14.07 ± 0.20	1.00	14.33 ± 0.22	1.00	13.74 ± 0.12	1.00	13.81 ± 0.18	1.00
	Heterogenous	15.00 ± 0.20	1.00	13.82 ± 0.17	1.00	14.82 ± 0.21	1.00	14.73 ± 0.22	1.00	14.57 ± 0.23	1.00
	Control	11.30	1.00	13.47	1.00	12.85	1.00	10.20	1.00	12.19	1.00
Label	Homogeneous	4.37 ± 2.93	0.29 ± 0.26	4.33 ± 2.93	0.30 ± 0.25	4.45 ± 3.13	0.32 ± 0.27	4.38 ± 3.07	0.29 ± 0.26	4.44 ± 3.15	0.31 ± 0.27
	Heterogenous	3.15 ± 2.37	0.12 ± 0.26	3.20 ± 2.44	0.14 ± 0.34	3.26 ± 2.62	0.13 ± 0.27	3.12 ± 2.36	0.14 ± 0.26	3.41 ± 2.82	0.16 ± 0.24
$t = 0.1$											
	Control	79.98	1.00	81.51	1.00	83.88	1.00	83.04	1.00	84.18	1.00
Noise	Homogeneous	61.71 ± 3.23	1.00	59.94 ± 3.55	1.00	58.78 ± 2.77	1.00	61.51 ± 2.44	1.00	62.14 ± 2.65	1.00
	Heterogenous	61.43 ± 3.17	1.00	62.16 ± 2.75	1.00	62.83 ± 2.81	1.00	62.39 ± 3.57	1.00	60.14 ± 3.01	1.00
	Control	11.75	1.00	14.91	1.00	13.61	1.00	10.84	1.00	13.49	1.00
Label	Homogeneous	4.37 ± 2.93	0.29 ± 0.26	4.33 ± 2.93	0.30 ± 0.25	4.45 ± 3.13	0.32 ± 0.27	4.38 ± 3.07	0.29 ± 0.26	4.44 ± 3.15	0.31 ± 0.27
	Heterogenous	3.15 ± 2.37	0.12 ± 0.26	3.20 ± 2.44	0.14 ± 0.34	3.26 ± 2.62	0.13 ± 0.27	3.12 ± 2.36	0.14 ± 0.26	3.41 ± 2.82	0.16 ± 0.24
$t = 0.01$											
	Control	186.03	1.00	186.13	1.00	188.30	1.00	184.57	1.00	187.54	1.00
Noise	Homogeneous	74.38 ± 5.80	0.99	73.04 ± 6.59	0.99	74.09 ± 6.37	0.99	73.81 ± 5.72	0.99	74.51 ± 5.97	0.99
	Heterogenous	72.94 ± 3.51	0.99	72.34 ± 3.80	0.99	73.78 ± 4.50	0.99	73.50 ± 4.22	0.99	72.12 ± 6.00	0.99
	Control	11.77	1.00	15.07	1.00	13.66	1.00	10.90	1.00	13.60	1.00
Label	Homogeneous	4.37 ± 2.93	0.29 ± 0.26	4.33 ± 2.93	0.30 ± 0.25	4.45 ± 3.13	0.32 ± 0.27	4.38 ± 3.07	0.29 ± 0.26	4.44 ± 3.15	0.31 ± 0.27
	Heterogenous	3.15 ± 2.37	0.12 ± 0.26	3.20 ± 2.44	0.14 ± 0.34	3.26 ± 2.62	0.13 ± 0.27	3.12 ± 2.36	0.14 ± 0.26	3.41 ± 2.82	0.16 ± 0.24