Beyond Aggregation: Efficient Federated Model Consolidation with Heterogeneity-Adaptive Weights Diffusion

Jiaqi Li[1] Siqi Ping[2]

Abstract

With the rapid development of the Internet of Things (IoT), the need to expand the amount of data through data-sharing to improve the model performance of edge devices has become increasingly compelling. To effectively protect data privacy while leveraging data across silos, Federated Learning (FL) has emerged. However, communication costs in FL constitute a critical bottleneck for broader practical applications. Traditional approaches like Federated Averaging (FedAvg) and recent methods targeting increased communication efficiency primarily focus on client-side enhancements, leaving server-side aggregation largely untapped, potentially exacerbating client-side computations. Along with diffusion models shining in AIGC, we pioneer their use for model weights generation in FL's server component, substituting conventional aggregation algorithms. We propose a novel federated model consolidation way with heterogeneity-adaptive weights diffusion(**FedDiff**) for communication efficient FL. Rigorous experimentation verifies that our method delivers exceptional convergence speed and accuracy, and exhibits robustness against weights noise.

1. Introduction

Federated Learning (FL) (Kairouz et al., 2021) has emerged as a promising paradigm for training machine learning models in a distributed manner, enabling collaborative learning across multiple clients, such as mobile devices (Lim et al., 2020), without centralizing data. This is particularly advantageous for privacy-preserving applications where data sensitivity precludes aggregation (Truex et al., 2019; Zhu et al., 2019). However, FL faces a significant challenge in the form of communication costs, which can be prohibitive, especially when the number of participating clients is large or the models are complex (Sattler et al., 2019; Zhao et al., 2023).

Traditional FL approaches, such as Federated Averaging (FedAvg) (McMahan et al., 2017), have made strides in reducing communication overhead by averaging local model updates. However, these methods are primarily focused on client-side optimizations (Li et al., 2020) and do not fully explore potential server-side enhancements. The server, central to aggregating client updates (Sun et al., 2024; Hsu et al., 2019), presents an untapped opportunity for innovation that could significantly reduce the communication bottleneck. This raises a crucial question: *Can we find a new method to aggregate knowledge on the server side?*

Diffusion Model for Weights Generation. Diffusion models (Sohl-Dickstein et al., 2015) have achieved remarkable results in visual generation. These methods (Ho et al., 2020; Nichol & Dhariwal, 2021; Song & Ermon, 2019; Ramesh et al., 2021) are based on non-equilibrium thermodynamics and are akin to GANs (Goodfellow et al., 2020), VAEs (Variational Autoencoders) (Kingma & Welling, 2013), and flow-based models (Dinh et al., 2014). Diffusion models can be categorized into three main branches: enhancing synthesis quality (e.g., DALL.E 2 (Ramesh et al., 2022), Imagen (Saharia et al., 2022), and Stable Diffusion (Rombach et al., 2022)), improving sampling speed (e.g., DDIM (Song et al., 2020), Analytic-DPM (Bao et al., 2022), and DPM-Solver (Lu et al., 2022)), and reevaluating diffusion models from a continuous perspective, like score-based models (Song & Ermon, 2019).

Different from these branches, the use of diffusion for weight generation is gradually becoming practical and promising. Works in this field, such as G.pt (Peebles et al., 2022), MetaDiff (Zhang et al., 2024), Hyperdiffusion (Erkoç et al., 2023), P-diff (Wang et al., 2024), and ProtoDiff (Du et al., 2023), demonstrate that diffusion models can generate model weights with better accuracy than the SGD (Stochastic Gradient Descent) optimizer (Bottou, 1998). Remarkably, diffusion models can generate model weights in finite time steps, bypassing the need for extensive SGD training. Additionally, the federated process mirrors the denoising process in diffusion models: both involve transitions from

[1]Shenzhen International Graduate School, Tsinghua University, China [2]Data Science and Information Technology Research Center, Tsinghua-Berkeley Shenzhen Institute, Tsinghua University, China.

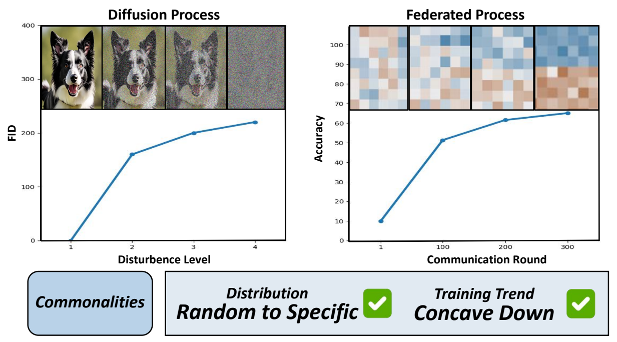


Figure 1. The left panel illustrates the standard diffusion process in image generation. The right panel shows the parameter distribution of batch normalization (BN) during the training of CIFAR-10 with ResNet-18 in a federated setting.

random noise/initialization to specific distributions, and both training trends are concave functions (illustrated in Fig. 1). Therefore, an intuitive idea is to harness the remarkable and efficient generative capability of diffusion models to help consolidate model weights. To put it another way, we aim to avoid the numerous communication rounds in heterogeneous FL, just as the diffusion model directly generates model weights, bypassing hundreds or even thousands of times of SGD training.

This idea poses two challenges: i) *How to aggregate models with the assistance of the diffusion model?* ii) *How to maintain the applicability of the diffusion model in heterogeneous FL?* To address these challenges, we introduce a novel approach to server-side optimization in FL by leveraging VAEs and weights-generated diffusion models. Our key insight is to train a VAE encoder coupled with a pre-trained diffusion model on the server to generate consolidated model weights, leading to efficient FL and replacing the standard parameter aggregation scheme.

Our contributions are two-fold:

- We innovatively employ VAEs and diffusion models for weight generation in the context of FL, achieving improvements in both communication costs and accuracy over traditional aggregation methods.

- We introduce KL divergence loss and contrastive loss for latent representation to fine-tune the encoder, ensuring the applicability of the diffusion model in heterogeneous FL.

2. Approach

2.1. Preliminaries

Diffusion Model for Weights Generation. Based on the exploration of previous related work, training a diffusion model directly in the parameter space tends to have a relatively high memory cost, considering the dimension of the parameter model that needs to be generated. Therefore, taking this into account, these studies apply the diffusion process to the latent representations by default. That is, let's assume the model is $\theta \in \mathbb{R}^D$, and D denotes the size of the model weights. And we first map the model weights to the latent space, denoted as Z. Subsequently, the following forward and reverse processes are carried out.
Forward Process. The forward diffusion process involves successive Gaussian noise perturbations of Z over T time steps. At time step t,

$$p\left(Z^t \mid Z^{t-1}\right) = \mathcal{N}\left(Z^t; \mu_t = \sqrt{1-\beta_t} Z^{t-1}, \beta_t I\right), \quad (1)$$

where $\beta_t \in (0,1)$ is the noise variance.
Reverse Process. As in most DDPM approaches the reverse process is approximated by a neural network such that:

$$p_\psi\left(Z^{t-1} \mid Z^t\right) = \mathcal{N}\left(Z^{t-1}; \mu_\psi\left(Z^t, t\right), \Sigma_\psi\left(Z^t, t\right)\right), \quad (2)$$

where μ_ψ and Σ_ψ are neural networks parameterized by ψ.
Problem Definition. In federated learning, we solve an optimization problem of the form:

$$\min_{\theta \in \mathbb{R}^D} f(\theta) = \frac{1}{m} \sum_{i=1}^{m} F_i(\theta), \quad (3)$$

where θ denotes the model weights and $F_i(\theta) = \mathbb{E}_{\xi \sim \mathcal{D}_i}\left[f_i(\theta, \xi)\right]$ is the loss function of the i^{th} client and $\mathcal{D}_i$ is the data distribution for the i^{th} client. For $i \neq j, \mathcal{D}_i$ and $\mathcal{D}_j$ may be very different, and it is referred to as the heterogeneous data setting. In this paper, $\{\mathcal{D}_i\}_{i=1}^N$ are not identical.

A common approach to solving Eq.(3) in federated settings is FedAvg (McMahan et al., 2017). At each round of FedAvg, a subset of clients is selected (typically randomly) and the server broadcasts its global model to each client. In parallel, the clients run SGD on their own loss function and send the resulting model to the server. The server then updates its global model to match the average of these local models. Suppose that at round r, the server has model θ^r and samples a set $\mathcal{S}$ of clients. Let θ_i^r denote the model of each client $i \in \mathcal{S}$ after local training. We rewrite FedAvg's update as:

$$\theta^{r+1} = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \theta_i^r = \theta^r - \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \left(\theta^r - \theta_i^r\right). \quad (4)$$

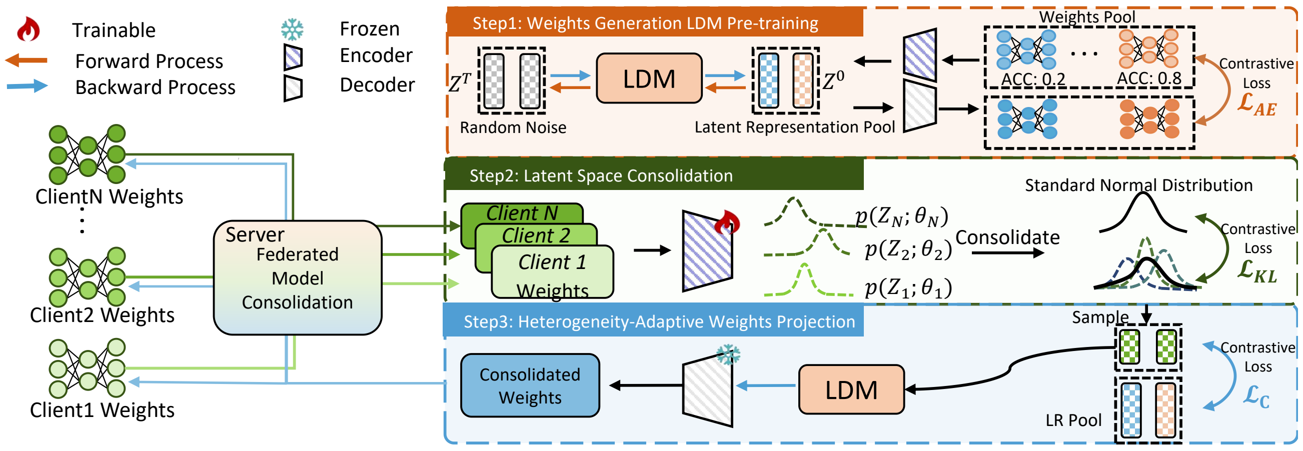


Figure 2. The FedDiff framework encompasses three steps. It utilizes three contrastive loss, with $\mathcal{L}_{AE}$ designated for the pre-training of the autoencoder (AE), and $\mathcal{L}_{KL}$ along with $\mathcal{L}_C$ for the fine-tuning of the pre-trained encoder.

The Eq.(4) is a naive aggregation algorithm on the server. This paper focuses on the server side's aggregation or optimization way. That is, it aims to use the efficient federated model consolidation to substitute the naive federated aggregation.

2.2. Neural Network Weights Generation Diffusion model Pre-training

In this section, the paper primarily utilizes the concept of P-diff (Wang et al., 2024). A subset of the data is selected from the dataset to serve as a proxy dataset; these data points are uniformly sampled and can be considered as visible data shared by both the client and the server. We train the model ResNet-18 (He et al., 2016) using these data with SGD optimizer to obtain the model weights. The weights from steps that are multiples of 10 are retained and then sorted by accuracy from low to high, forming a pool of weights. We use the weights pool to train an Autoencoder (AE) that can obtain latent representations corresponding to different accuracies and training epochs, as well as a Latent Diffusion Model (LDM) that can output latent representations with varying accuracies based on different time steps T.

Weight Encoding. We flatten these weights $\theta \in \mathbb{R}^D$ into 1-dim vector $V = [v_1, \ldots, v_i, \ldots, v_W]$, where $V \in \mathbb{R}^{D\times W}$ and W is the size of the weights pool. After that, an AE is trained to reconstruct these weights V. To enhance the robustness and generalization of the AE, we introduce random noise augmentation. The encoding and decoding processes can be formulated as:

$$Z^0 = \underbrace{g_\sigma(V + \xi_V)}_{\text{encoding}}; V' = \underbrace{g_\rho(Z^0 + \xi_Z)}_{\text{decoding}}, \tag{5}$$

where $g_{\text{encoder}}(\cdot, \sigma)$ and $g_{\text{decoder}}(\cdot, \rho)$ denote the encoder and decoder parameterized by σ and ρ, respectively. Z^0 represents the latent representations, ξ_V and ξ_Z denote random noise that are added into input weights V and latent representations Z^0, and V' is the reconstructed weights. we minimize the mean square error (MSE) loss between V' and V as follows:

$$\mathcal{L}_{AE} = \frac{1}{W}\sum_{i=1}^{W} \left\| v_i - v_i' \right\|^2, \tag{6}$$

where v_k' is the reconstructed weights of k-th model.

Diffusion Model Pre-training. At this stage, we achieve defining LDM to generate latent representations of weights by using DDPM (Ho et al., 2020). The diffusion model is trained on the AE embeddings Z^0 with different time steps T. To take advantage of existing architectures, we use minimal modifications and optimize the following latent diffusion model objective:

$$\mathcal{L}_{LDM} = \mathbb{E}_{Z,\varepsilon\sim\mathcal{N}(0,1),t}\left[\left\|\varepsilon - \varepsilon_\psi\left(Z^t, t\right)\right\|^2\right], \tag{7}$$

where $\varepsilon_\psi\left(Z^t, t\right)$ is implemented as a 1D-CNN.

2.3. Federated Model Consolidation

Latent Space Consolidation. At this stage, we aim to consolidate the weights of the sampled clients and we have access to a pre-trained AE and diffusion model that could generate the weights from the noise. Assuming that at round r, the server has model θ^r and samples a set $\mathcal{S}$ of clients. Let θ_i^r denote the model of each client $i \in \mathcal{S}$ after local training. We use the pre-trained encoder to encode the weights θ_i^r. Therefore, after we get the latent representation $Z_i = g_\sigma(\theta_i^r)$, we consolidate the knowledge of sampled clients.

Through repeated experiments and observations, it has been found that the normal federated aggregation way leads to the weight distributions of the weights of the model varying tangibly with more weights concentrated on 0. So we propose a novel method for generating new global latent representations from the sampled clients. The essential but simple method includes two parts: i) the distribution superiposition; and ii) the reparameterization trick. Let Z_i be a continuous random variable, and Z_i be the normal distribution based on our prior knowledge. The global Z_s is sampled from the superimposed normal distribution. It is then often possible to express the random variable Z_s as a deterministic function $Z_s = g_\sigma(\epsilon, \theta_i; i \in \mathcal{S})$, where ϵ is an auxiliary variable with independent marginal $p(\epsilon)$, and $g_\sigma(\cdot)$ is the encoding vector-valued function parameterized by σ defined in Eq.(5).

More specifically, let $Z_i \sim p(Z_i|\theta_i) = \mathcal{N}(\mu_i, \sigma_i^2)$, and we assume the normal distributions are independent. Therefore, we consolidate independent normal distribution and approximate the mean and variance of the consolidated distribution in a simplified way:

$$\mu_s = \frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\mu_i; \sigma_s^2 = \frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\sigma_i^2. \tag{8}$$

Next, we sample the consolidated latent representation using the mean and variance calculated above, it can be approximated that the superimposed distribution is also a normal distribution $\mathcal{N}(\mu_s, \sigma_s^2)$. Then, sampling can be carried out through the following steps: i) generate random samples $\epsilon^{(i)}$ from the standard normal distribution $\mathcal{N}(0,1)$, and ii) utilize the reparameterization technique to convert the samples from the standard normal distribution into samples from the superimposed distribution: $Z_s = [z_1, \ldots, z_i, \ldots, z_{|\mathcal{S}|}]$, where $z_i = \mu_s + \sigma_s \cdot \epsilon^{(i)}$. Additionally, to constrain the distribution of our consolidated latent space, we draw on the ideas from VAEs and modify the MSE loss function in Eq.(6):

$$\mathcal{L}_{KL} = \frac{1}{|S|}\sum_{i=1}^{|S|}\|v_i - v_i'\|^2 + \alpha KL[\mathcal{N}(\mu_s, \sigma_s^2)||\mathcal{N}(0,1)]. \tag{9}$$

Heterogeneity-Adaptive Weights Projection. Taking into account that the weights we used during the pre-training process were trained on datasets that were divided in an independent and identically distributed (iid) manner. However, in federated aggregation, the data from different clients is actually non-iid. Therefore, in this section, we innovatively choose to fine-tune the pre-trained encoder to complete the projection of the weights from non-iid to iid to ensure the applicability of the pre-trained diffusion model in heterogeneous FL. Our approach primarily aims to ensure that after the model weights from different clients is projected and then sampled, the latent space is as close as possible to the latent representation pool when trained with iid. Therefore, we introduce the contrastive loss for latent representation.

That is, for a consolidated representation Z_s with a certain accuracy, we select the LR from the LR pool that is similar in accuracy to serve as the corresponding LR Z_{corr}. We compare the latent representation Z_s with the one trained using SGD as follows:

$$\mathcal{L}_C = \frac{1}{|\mathcal{S}|}\sum_{i=1}^{|\mathcal{S}|}||z_{corr} - z_i||^2. \tag{10}$$

Up to here, we can get the loss of the encoder as follows:

$$\mathcal{L}_{Encoder} = \mathcal{L}_{KL} + \beta\mathcal{L}_C, \tag{11}$$

and we change the naive aggregation method Eq.(4) to the consolidated approach:

$$\theta^{t+1} = g_\rho(p_\psi(Z_s))\prod_{t=1}^{T}p_\psi(Z_s^{t-1}|Z_s^t). \tag{12}$$

where Z_s^0 denotes the consolidated weights output by pre-trained latent diffusion model. Considering that the LDM can output models with varying accuracies, as the number of communication rounds r increases, the time steps $T \propto r$ of the LDM.

3. Experiments

In this section, we present empirical evidence to verify our framework. We train ResNet-18 on MNIST (LeCun et al., 1998), CIFAR-10/100 (Krizhevsky et al., 2009) and TinyImageNet (Tavanaei, 2020).

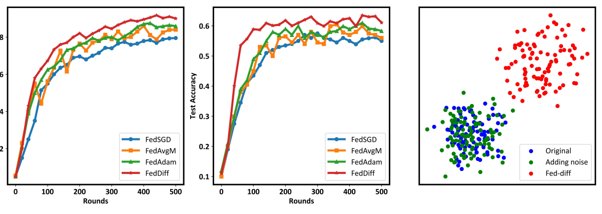
3.1. Experimental Setting

Federated Setting. To simulate data heterogeneity, we impose label imbalance across 10 clients, i.e. each client is allocated a proportion of the samples of each label according to a Dirichlet distribution with the concentration parameter $\alpha = 0.5$. The partial participation ratio is 0.5, i.e., 5 out of 10 clients are picked in each round, and $|\mathcal{S}| = 5$. The local epoch K is 100.
Diffusion Setting. We set the dataset partition ratio is 0.04, i.e., 2000 out of 50000 images from CIFAR-10 are selected for the weights pool.We set the total diffusion time steps $T = 200$, and the selected images are further divided into 20 parts, each part is used for training ResNet-18 for 100 to 2000 times using SGD optimizer.
Compared Methods. We use the FedAvg (McMahan et al., 2017) as our baseline. And we use the following two-category works as our compared methods: i) Server Momentum Methods: FedAvgM (Hsu et al., 2019), SCAFFOLD (Karimireddy et al., 2020), STEM (Khanduri et al., 2021), FedGM (Sun et al., 2024) and ii) Adaptive Methods: FedAdam (Reddi et al., 2020), FedAMS (Wang et al., 2022), FAFED (Wu et al., 2023).

Table 1. non-IID setting. Test accuracy and the number of communication rounds required for each method to achieve 95% of its respective highest metric. over 10 clients on the MNIST, CIFAR-10, CIFAR-100, and TinyImageNet datasets.

	MNIST		CIFAR-10		CIFAR-100		TinyImageNet	
Methods	Top-1 Acc	Rounds	Top-1 Acc	Rounds	Top-5 Acc	Rounds	Top-5 Acc	Rounds
FedSGD	86.46 ± 4.02	483	49.40 ± 2.19	605	49.98 ± 1.10	560	35.75 ± 0.56	711
FedAVG	91.42 ± 2.31	334(1.4×)	54.12 ± 3.05	390(1.6×)	55.71 ± 0.35	710(0.8×)	41.61 ± 3.59	664(1.1×)
FedAvgM	92.20 ± 1.55	219(2.2×)	57.42 ± 2.45	319(1.9×)	54.45 ± 1.77	416(1.4×)	43.88 ± 2.47	549(1.3×)
SCAFFOLD	92.51 ± 1.18	246(2.0×)	57.63 ± 1.72	254(2.4×)	56.70 ± 1.19	409(1.4×)	44.34 ± 2.01	531(1.3×)
STEM	91.13 ± 1.49	194(2.5×)	59.40 ± 2.79	276(2.2×)	55.29 ± 2.37	281(2.0×)	45.48 ± 0.89	310(2.3×)
FedGM	93.76 ± 0.96	153(3.2×)	58.20 ± 1.72	301(2.0×)	56.11 ± 1.08	398(1.4×)	47.70 ± 2.99	433(1.6×)
FedAdam	93.15 ± 2.06	114(4.2×)	59.87 ± 2.31	313(1.9×)	54.31 ± 2.70	339(1.7×)	45.29 ± 1.09	462(1.5×)
FedAMS	90.92 ± 2.64	166(2.9×)	57.40 ± 1.57	255(2.4×)	54.38 ± 1.98	200(2.8×)	44.73 ± 1.38	411(1.7×)
FAFED	94.43 ± 1.80	174(2.8×)	56.07 ± 3.78	206(2.9×)	57.44 ± 3.83	343(1.6×)	46.73 ± 1.38	384(1.9×)
FedDiff (ours)	94.50 ± 2.96	73(6.6×)	62.57 ± 3.78	107(5.7×)	57.97 ± 1.74	132(4.3×)	51.60 ± 3.91	224(3.2×)

Training Details. The autoencoder and latent diffusion model both include a 4-layer 1D CNNs-based encoder and decoder. The ξ_V and ξ_Z are Gaussian noise with amplitudes of 0.01 and 0.1. We set the local learning rate is 0.001 and the batch size is 64. In most cases, the whole training includes the pre-training and federated learning can be completed within 4 hours on a single Nvidia A100 40G GPU.



(a) Train accuracy on CIFAR-10 (b) Test accuracy on CIFAR-10 (c) t-SNE of the latent representations

Figure 3. (a) and (b) shows the training and testing curves for FedDiff (ResNet-18 on CIFAR-10). (c) displays the t-SNE of the original models, models after the federated consolidation, and adding noise operation.

3.2. Result Analysis

The goal of our experiments is four-fold: i). To compare the performance of FedDiff with other algorithms in heterogeneity setting during the training phase with training datasets; ii). To demonstrate the model performance on the test datasets; iii). To compare the number of communication rounds required for each method to achieve 95% of its respective highest metric; and iv). To verify the applicability of the diffusion model in FedDiff by comparing the latent representaion using t-SNE (Van der Maaten & Hinton, 2008). Fig.3 (a) and (b) shows the results for ResNet-18 on CIFAR-10 with FedDiff, FedSGD, FedAvgM and FedAdam. We observe that though FedAvgM converges faster than FedSGD, it is only marginally better in terms of testing. FedDiff, in contrast, outperforms FedAvgM and FedSGD in both measures. Moreover, it is worth noting that during the testing phase, FedDiff has the fastest convergence rate, achieving a considerably high accuracy around the 100^{th} epoch. *The main difference between FedDiff and other methods lies in the early stages of federated learning, where the convergence speed is significantly accelerated with the help of the diffusion model.* Table. 1 also proves this point. In comparisons with more methods and more datasets, due to the accelerated convergence speed in the early stages of training, FedDiff has shown several times the convergence rate of existing methods and the fastest convergence rate achieving 95% of its highest metric compared to other methods. Except for the above results, Fig.3 (c) proves the applicability of the weights generation LDM in heterogenous FL by comparing the distributions of the latent representations for the original and consolidated models. We can observe that FedDiff is capable of generating novel latent representations even in the presence of weights noise introduced by misclassified client data and the gradient noise during the communication process in real FL. This means it can maintain robustness against weights noise.

4. Conclusion

In summary, this paper introduces the brave idea of integrating diffusion models into FL to enhance server-side optimization. By designing the federated model consolidation framework, we have harnessed the powerful and finite-step weight generation capability of LDM to accelerate federated learning, especially in its early stages. This approach reduces communication overhead while maintaining commendable performance and has potential applications in a variety of privacy-sensitive domains.

5. Author Contribution

Jiaqi Li built the model and ran part experiments. Siqi Ping ran part experiments, carried out the poster creation and report transcription.

References

Bao, F., Li, C., Zhu, J., and Zhang, B. Analytic-dpm: an analytic estimate of the optimal reverse variance in diffusion probabilistic models. *arXiv preprint arXiv:2201.06503*, 2022.

Bottou, L. Online algorithms and stochastic approximations. *Online learning in neural networks*, 1998.

Dinh, L., Krueger, D., and Bengio, Y. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Du, Y., Xiao, Z., Liao, S., and Snoek, C. Protodiff: Learning to learn prototypical networks by task-guided diffusion. *Advances in Neural Information Processing Systems*, 36: 46304–46322, 2023.

Erkoç, Z., Ma, F., Shan, Q., Nießner, M., and Dai, A. Hyperdiffusion: Generating implicit neural fields with weight-space diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14300–14310, 2023.

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

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Hsu, T.-M. H., Qi, H., and Brown, M. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2):1–210, 2021.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., and Suresh, A. T. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, pp. 5132–5143. PMLR, 2020.

Khanduri, P., Sharma, P., Yang, H., Hong, M., Liu, J., Rajawat, K., and Varshney, P. Stem: A stochastic two-sided momentum algorithm achieving near-optimal sample and communication complexities for federated learning. *Advances in Neural Information Processing Systems*, 34: 6050–6061, 2021.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

Lim, W. Y. B., Luong, N. C., Hoang, D. T., Jiao, Y., Liang, Y.-C., Yang, Q., Niyato, D., and Miao, C. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 22(3): 2031–2063, 2020.

Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022.

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

Nichol, A. Q. and Dhariwal, P. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pp. 8162–8171. PMLR, 2021.

Peebles, W., Radosavovic, I., Brooks, T., Efros, A. A., and Malik, J. Learning to learn with generative models of neural network checkpoints. *arXiv preprint arXiv:2209.12892*, 2022.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. Zero-shot text-to-image generation. In *International conference on machine learning*, pp. 8821–8831. Pmlr, 2021.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.

Reddi, S., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečný, J., Kumar, S., and McMahan, H. B. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E. L., Ghasemipour, K., Gontijo Lopes, R., Karagol Ayan, B., Salimans, T., et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35: 36479–36494, 2022.

Sattler, F., Wiedemann, S., Müller, K.-R., and Samek, W. Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural networks and learning systems*, 31(9):3400–3413, 2019.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. PMLR, 2015.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Sun, J., Wu, X., Huang, H., and Zhang, A. On the role of server momentum in federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 15164–15172, 2024.

Tavanaei, A. Embedded encoder-decoder in convolutional networks towards explainable ai. *arXiv preprint arXiv:2007.06712*, 2020.

Truex, S., Baracaldo, N., Anwar, A., Steinke, T., Ludwig, H., Zhang, R., and Zhou, Y. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM workshop on artificial intelligence and security*, pp. 1–11, 2019.

Van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Wang, K., Xu, Z., Zhou, Y., Zang, Z., Darrell, T., Liu, Z., and You, Y. Neural network diffusion. *arXiv preprint arXiv:2402.13144*, 2024.

Wang, Y., Lin, L., and Chen, J. Communication-efficient adaptive federated learning. In *International Conference on Machine Learning*, pp. 22802–22838. PMLR, 2022.

Wu, X., Huang, F., Hu, Z., and Huang, H. Faster adaptive federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 10379–10387, 2023.

Zhang, B., Luo, C., Yu, D., Li, X., Lin, H., Ye, Y., and Zhang, B. Metadiff: Meta-learning with conditional diffusion for few-shot learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 16687–16695, 2024.

Zhao, Z., Mao, Y., Liu, Y., Song, L., Ouyang, Y., Chen, X., and Ding, W. Towards efficient communications in federated learning: A contemporary survey. *Journal of the Franklin Institute*, 360(12):8669–8703, 2023.

Zhu, L., Liu, Z., and Han, S. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.