Towards Privacy-Preserving Split Learning for ControlNet

Dixi Yao
University of Toronto, Hemlig AI, EV.com
dixi.yao@mail.utoronto.ca, Dixi.y@ev.com

Abstract

With the emerging trend of large generative models, ControlNet is introduced to enable users to fine-tune pre-trained models with their own data for various use cases. A natural question arises: how can we train ControlNet models while ensuring users' data privacy across distributed devices? We first propose a new distributed learning structure that eliminates the need for the server to send gradients based on split learning. We discover that in the context of fine-tuning ControlNet with split learning, most existing attacks are ineffective, except for two mentioned in previous literature. To counter these threats, we leverage the properties of diffusion models and design a new timestep sampling policy during forward processes. We also propose a privacy-preserving activation function and a method to prevent private text prompts from leaving clients, tailored for image generation with diffusion models. Our experimental results demonstrate that our algorithms and systems greatly enhance the efficiency of distributed fine-tuning for ControlNet while ensuring users' data privacy without compromising image generation quality.

1. Introduction

Leading at the forefront in the emerging trend of large generative artificial intelligence, large diffusion models [35] have become commercial success stories, with models from Stability AI and Midjourney dominating the news. With large diffusion models, any user is able to generate artistically appealing images with short descriptive text prompts. However, short descriptive text prompts do not offer a sufficient level of control over the generated images to satisfy a user's needs in many cases. To support an additional level of control using *conditions*, ControlNet [45] has recently emerged, allowing users to generate images with a wide variety of user-defined conditions beyond text prompts.

With fine-grained control over generated images using ControlNet, it's intuitive that users would want to fine-tune pre-trained ControlNet models with their own data to meet various use cases. However, since the fine-tuning dataset may contain users' own artistic creations or faces, privacy concerns arise. Additionally, each user may possess only a small number of images, which may not suffice for fine-tuning a diffusion model unless aggregated, such as in a collection of *50,000* images [45]. To maintain data privacy, it's essential to fine-tune ControlNet with distributed users, posing the research question: How can we train ControlNet models while preserving users' data privacy, particularly when the data is distributed across multiple client devices?

Split learning [11] (SL) has been heralded in recent years as a distributed training paradigm that preserves user privacy. It is suitable for cases where clients lack substantial memory for local fine-tuning tasks. Split learning has a wide range of applications in various areas, allowing multiple clients with limited resources to collaboratively fine-tune a deep learning model, for example, in health care [38]. This expands the use cases for leveraging split learning to fine-tune large diffusion models such as ControlNets and other diffusion models.

In split learning, clients train the first few layers of the neural network with their local data and transmit *intermediate features* to the server. The server then sequentially sends gradients back to the clients after the forward pass and backpropagation. However, recent literature highlights that split learning can be inefficient and vulnerable to adversarial attacks, such as inversion attacks [10, 23, 39, 43], which have the potential to reconstruct private data.

With our analysis of existing attacks, we find that inversion attacks using inverse network models are effective for reconstructing conditional images when we train models with split learning. These attacks first train an inverse network on a public dataset and then use it to reconstruct private data [39, 43]. Furthermore, we find that defending against such successful attacks with existing defense mechanisms greatly degrades image generation performance.

Our original contributions are as follows:

First, to enhance the efficiency of fine-tuning ControlNets using split learning, we design a new structure, eliminating the need for the server to send data back to the clients, thereby addressing the issue of efficiency bottlenecks.

Second, inspired by our empirical observations, we find that the forward process when training diffusion models can be combined with local differential privacy guarantees. Based on this, we emphasize our privacy-preserving timestep scheduling policy, establishing a relationship between timestep scheduling and the privacy budget ϵ. This allows us to adjust the privacy-preserving ability of the system by setting specific scheduling policies. Additionally, we propose a symmetric activation function to process intermediate features, preventing attackers from reconstructing condition images while still generating high-quality images.

Third, in addition to the privacy leakage of conditional images, we further explore the leakage of text prompts. To train the diffusion model and ControlNet, we need to upload the text prompts, which may contain private information, to the server. We propose a new mechanism fine-tuning ControlNets with zero prompts. The fine-tuned model maintains high performance in image generation, while the server does not know the text prompts.

Finally, to evaluate performance fairly, we implement a system to train ControlNet with split learning using PLATO. It is demonstrated that with our architecture design, clients require less than 3 GB of GPU memory and experience $3\times$ lower communication overhead. Unlike existing privacy-preserving mechanisms, we verify that our mechanism can protect the privacy of images, conditions, and text prompts without sacrificing image generation performance. The code is available at `https://github.com/TL-System/plato/tree/main/examples/split_learning/controlnet_split_learning`.

2. Background and Related Work

2.1. Diffusion Model and ControlNet

Diffusion Models [35] are probabilistic models gradually denoising a normally distributed variable to generate high-quality images. Existing diffusion models allow users to guide image generation with text prompts. It is common in diffusion model [33] to convert images into latent representations with an encoder $\mathcal{E}$ and conducts the diffusion process on the latent domain Z. After the sampling process, images are generated through a corresponding decoder $\mathcal{D}$.

The image generation process involves a sampling procedure, which is the inverse of the forward process. In the forward process, we follow a Markov Chain to gradually add Gaussian noise ($\mathcal{N} \sim (0, 1)$) to the data, based on a variance schedule $\beta_1, \dots, \beta_T$, where $t \in [T]$ represents each timestep of noise addition. We denote this Gaussian noise as $\hat{n}$. As an inverse of the diffusion process, during the sampling process, the diffusion model outputs an estimation of noise n at timestep t, and we sample the latent z_{t-1} using the equation:

$$
\begin{aligned}
z_{t-1} =& \sqrt{\alpha_{t-1}} \left(\frac{z_t - \sqrt{1-\alpha_t} n}{\sqrt{\alpha_t}} \right) \\
&+ \sqrt{1 - \alpha_{t-1} - \lambda_t^2} n + \lambda_t o(z_t)
\end{aligned}
\tag{1}
$$

Here, $\alpha_t = 1 - \beta_t$, λ_t is a noise coefficient, and $o(z_t)$ is randomly generated from a standard normal distribution. The sampling process begins with Gaussian noise and gradually samples until obtaining z_0, corresponding to the latent representation of the image to be generated.

In each training step, we follow Eq. (1) to generate random noise $\hat{n}$ as a label, serving as the ground truth. The diffusion model's objective is to learn the parameters θ to infer the noise n. This inferred noise, the model's output, is used for denoising the image.

To enable users to control the generated images with more detailed conditions such as scribbles [45], canny lines [1], depth maps [32], HED lines [40], and segmentation maps [47], in addition to the given text prompts, a conditional diffusion model is proposed.

Apart from stable diffusion models [33], ControlNet can leverage other backbones such as LCM [24] and Control-LoRA [14]. Concurrent works, T2I-Adapter [27] and Composer [18], feature much smaller and larger control networks, respectively. FreeDoM [44] is a training-free conditional diffusion model. However, generating images with fine-grained conditions, such as canny edge maps, can be challenging, resulting in poor guidance. Training-required methods remain the optimal solution for conditional diffusion models.

2.2. Decentralized Training of ControlNet

With the assistance of ControlNet, users can fine-tune well-trained stable diffusion (SD) models without disrupting the original SD models. However, the conditions and training images involved may contain privacy-sensitive information. One straightforward solution is to train ControlNet entirely on a single device. For inference with a batch size of 1, we need 7.50 GB of GPU memory. However, to train ControlNet, a minimum of 23.82 GB of GPU memory is needed (with a minimal batch size of 2).

Even if a client has enough GPU memory to fine-tune a diffusion model locally, another issue arises when collecting training samples from different users, as it may lack sufficient data. To enable users to fine-tune ControlNets without their private data leaving their devices, a common solution is to use privacy-preserving decentralized frameworks.

There are also data encryption approaches in decentralized systems, such as trusted execution environments, multiparty computation, and homomorphic encryption. However, the overhead is not at the same scale as computing over plaintext data. For example, during inference, the forward time on diffusion models with homomorphic encryption [3] is 79.19 days, compared to 35 seconds with plaintext using NVIDIA A100. Moreover, to the best of our knowledge, there is no encryption method that can be directly applied to the training process of diffusion models.

To address training challenges of conditional diffusion models on clients or servers, split learning offers a viable approach, involving multiple clients and a server. Split learning spans a neural network across the cloud and edge. The edge device trains up to a partition layer and sends intermediate features to the server, which completes forward propagation with remaining layers. During back-propagation, the server propagates to the partition layer, sending gradients to the client, which updates local parameters. However, split learning's sequential nature leads to resource underutilization and high transmission overhead, extending training. Each step requires feature and gradient exchanges, with one party waiting as the other computes or transmits data.

2.3. Privacy Leakage in Split Learning

Potential threats arise from SL, as it carries the risk of privacy leakage through data transmission between clients and the server. Literature highlights that an honest-but-curious server could reconstruct private data using the intermediate features sent from clients to the server. Zhang et al. [46] successfully reconstructed private data with knowing model weights. UnSplit [10] further refined this method to conduct a similar attack without knowing model weights. He et al. [13] trained an inverse network using a public dataset, taking intermediate results as inputs to output private data for reconstruction. Pasquini et al. [29] proposed an attack to reconstruct private data by manipulating gradients sent back to clients, under the assumption of a dishonest server. Duan et al. [6] introduced a membership inference attack (MIA) tailored specifically for diffusion models, although they acknowledged its limited applicability in real-world scenarios. Additionally, Carlini et al. [2] utilized leaked text prompts to generate numerous images, subsequently employing MIA to identify which images exist in private datasets.

2.4. Privacy Protection in Split Learning

In response to potential privacy leakage in split learning, researchers have made efforts to defend against such attacks. Local differential privacy techniques, such as additive noise and randomized response [9], are employed to prevent reconstruction. Additionally, Gaussian noise is utilized to directly add noise to the raw data [23]. Subsequently, many works have adopted methods involving additive noise [26, 36, 37]. DataMix [23] and CutMix [28] leverage the concept of mixing a batch of samples. DataMix is designed for convolutional neural networks, while CutMix is tailored for vision image transformers. PatchShuffling [41–43] is a method specifically designed for transformer-structured models, where patches are shuffled among a batch of samples. However, all these methods must provide sufficient privacy guarantees at the cost of significant performance decreases.

Xiao et al. [39] utilized adversarial learning to enable clients to generate intermediate results that the server cannot use to reconstruct images. Shredder [26] introduced noise based on mutual information, while DISCO [34] employed

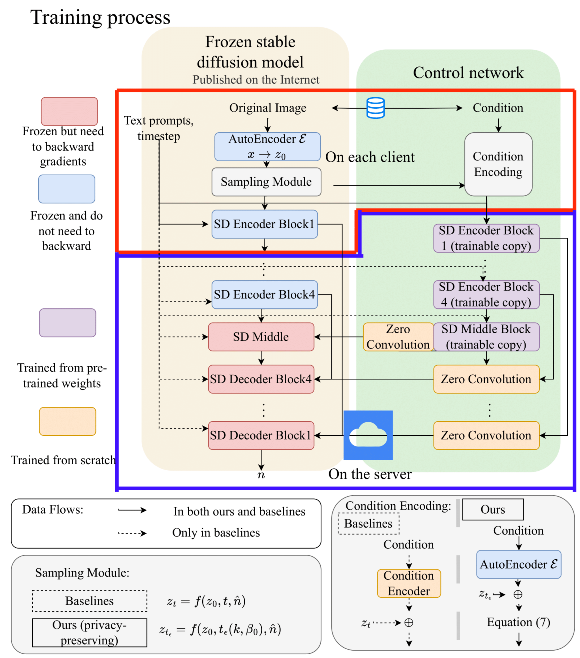

Figure 1. The deployment of ControlNet across the clients and the server under different structures. Function f is $z_t = f(z_0, t, \hat{n}) = \sqrt{\alpha_t} z_0 + \sqrt{1 - \alpha_t}\hat{n}$.

a channel obfuscation method to process features before transmitting them to the server. However, all three of these methods are only applied during the inference stage and require training a network to process features.

3. Speeding Up Fine-Tuning ControlNet

3.1. Fine-Tuning ControlNet with Split Learning

We first introduce the deployment for fine-tuning a diffusion model with ControlNet using split learning. Unlike dense models like ResNet [12], which have sequential blocks, the conditional diffusion model has two parts: a diffusion model with frozen weights and a trainable control network. The control network processes condition images with a condition encoder trained from scratch and mixes it with the noisy latent representation as input for the following blocks.

Considering hiding the complete model weights of the well-trained diffusion models from the clients and achieving the best tradeoff between privacy and efficiency through choosing different partition points, we cut right after the first diffusion model encoder and the trainable condition encoder.

Regarding the diffusion model, if we place the partition point before the first encoder block, the server can subtract the estimated noise n from received z_t in Eq. (1) to recover the z_0 and retrieve the private images.

3.2. Accelerating by Not Sending Gradients Back

To do split learning in practical use cases, we propose a new structure to address the efficiency bottleneck. This design ensures that the server does not need to send back gradients, thereby removing the sequential dependency between clients and the server during fine-tuning. Instead of training a condition encoder for each different condition, we propose to replace it with the pre-trained encoder used in the diffusion model. This way, clients only need to perform inference, allowing them to continuously forward without waiting for gradients from the server. This approach addresses the bottleneck caused by the sequential training manner.

As the clients share the same pre-trained model and the server model is shared between all clients, we do not need to aggregate client models. This makes the trained ControlNet have the same performance as centralized training. Besides that, since the condition encoder and pre-trained encoder both only need images as inputs, the replacement will not cause the outputs to have distribution drift. Hence, image generation performance will not be affected. We compare the memory usage, fine-tuning efficiency, and transmission overhead of these two structures in Tab. 1. The details about experimental settings are in Appendix A.

Without sending back the gradients, our new structure can save much transmission overhead. Additionally, by eliminating the forward-backward lock between the client and the server, the clients, server, and intermediate data transmission can operate in a parallel pipeline. Clients no longer need to wait for other clients or the server, reducing the time required for each client. Our whole fine-tuning time is $\max(\{\mathrm{T_c}, \mathrm{T_s}, \mathrm{T/r}\})$ while the original split learning structure needs $\mathrm{T_c} + \mathrm{T_s} + \mathrm{T/r}$ for fine-tuning, where r is the data transmission rate. We can increase the number of clients if we want, but since T_s is much larger than T_c, the whole fine-tuning time is the same.

4. Privacy-Preserving ControlNet Fine-Tuning

4.1. Threat Modeling

We begin by defining the threat model in practical scenarios. We assume the server to be honest but curious. In our designed split learning structure, although the server does not need to send gradients back to the clients, it will still accurately complete the remaining fine-tuning in each split learning iteration and send n to the clients. However, simultaneously, the server will attempt to reconstruct private data using the received intermediate features. The server can conduct the reconstruction process in the background, ensuring that clients remain unaware of the attacks. More details about the threats are presented in Appendix C.1.1.

Table 1. Comparison of memory usage, fine-tuning time, and transmission overhead for different structures. $\mathrm{M_c}$ and $\mathrm{M_s}$ denote GPU memory usage (in GB) on the client and server, respectively. $\mathrm{T_c}$ and $\mathrm{T_s}$ indicate fine-tuning time (in hours) on the client and server. T represents transmission overhead (in GB).

Structure	$\mathrm{M_c}$	$\mathrm{M_s}$	$\mathrm{T_c}$	$\mathrm{T_s}$	T
Split learning	2.78	22.04	22.46	14.10	559.17
Ours	2.75	22.04	0.446	14.10	186.56

4.2. Attacking Methods

Several threats in split learning include leakage of inputs to labels. The most threatening attack is the *inversion attack*, which attempts to reconstruct original private data from the received intermediate feature.

In this paper we will put emphasis on two threats. Inverse network-based attacks for reconstructing condition images and the leakage of text prompts. More details are presented in Appendix C.1.2 to show that these two threats are the remaining threats which are effective and valid in practical settings when fine-tuning ControlNets with split learning.

Inversion attack is based on training an inverse network [13, 23, 43]. In this approach, the attacker first trains an inverse network and it will take the intermediate features as inputs and outputs the reconstructed private data.

4.3. Local Differential Private Timestep Sampling

To achieve privacy protection against reconstructing private images, we can add noise to the original inputs or intermediate features [7], making it (ϵ, Δ)-LDP. The noise added can be Gaussian noise [5, 9].

Definition 4.1. ($(\epsilon, \Delta) - LDP$ noise adding.) A mechanism of adding noise over samples is $\epsilon-$LDP if the Gaussian noise follows the normal distribution

$$n \sim \mathcal{N}\left(0, 2\ln\frac{1.25}{\Delta}\alpha^2 \cdot \frac{1}{\epsilon^2}\right) \tag{2}$$

In literature, α is called sensitivity. It is the biggest L_2 distance between all possible inputs or intermediate features we are going to add noise.

If we examine the structure of ControlNet carefully, we find that the forward process involves adding noise to the latent representation. We will next show that this mechanism is $(\epsilon, \Delta)-$LDP using Definition 4.1. Based on this property, we propose a new sampling scheme over timesteps during the diffusion process, preserving privacy.

Given a latent representation z_0, we generate the noisy latent representation according to the timestep t, scheduling parameter β_t, and randomly generated noise $\hat{n}_t \sim \mathcal{N}(0,1)$:

$$z_t = \sqrt{1-\beta_t} z_0 + \sqrt{\beta_t}\hat{n}_t \Rightarrow \frac{z_t}{\sqrt{1-\beta_t}} = z_0 + \sqrt{\frac{\beta_t}{1-\beta_t}}\hat{n}_t \tag{3}$$

According to Fig. 1, z_t is the input to the first encoder block of diffusion model. Because β_t is usually a small number, we approximate Eq. (3) as,

$$z_t \approx z_0 + \sqrt{\frac{\beta_t}{1-\beta_t}}\hat{n}_t \tag{4}$$

We can view this equation as adding a noise following distribution $\mathcal{N}(0, \frac{\beta_t}{1-\beta_t})$ over z_0 to get z_t. We then substitute the variance in Definition 4.1. For convenience, we denote H as hyper-parameter $2\ln\frac{1.25}{\Delta}\alpha^2$.

$$\frac{\beta_t}{1-\beta_t} = H \cdot \frac{1}{\epsilon^2} \Rightarrow \epsilon = \sqrt{H \cdot \frac{1-\beta_t}{\beta_t}} \tag{5}$$

In the diffusion model, we employ the linear scheduling as the default method which is $\beta_t = k \cdot t + \beta_0$, where k, β_0 are scheduling parameters. Hence, we can derive the following relationship between privacy budget ϵ and t, k, and β_0:

$$\epsilon(t,k,\beta_0) = \sqrt{H \cdot \left(\frac{1}{kt+\beta_0} - 1\right)} \tag{6}$$

We can see that the privacy budget is related to the timestep. Based on Eq. (6), we can set proper privacy budget by setting different t, k, β_0 in fine-tuning ControlNet.

Remark 4.2. ((ϵ,Δ)-LDP timestep sampling mechanism in diffusion model) With a given privacy budget ϵ, we can have a sampling process in diffusion model which is (ϵ,Δ)-LDP. The value of ϵ is set by a timestep ranging in $[t_s, t_{\max}]$ and scheduling parameters k and β_0, according to Eq. (6).

4.4. Noise-Confounding Activation Function

However, as we can see from the Eq. (3), if we directly send the encoded condition mixed with the noisy latent representation to the server, as the server knows the timestep t and the label $\hat{n}$, it can directly subtract the added noise from Eq. (3). Hence, to confuse the attacker from stealing privacy, we propose to add a noise-confounding activation layer before sending features to the server. To design an activation function keeping privacy-preserving property while maintaining the image generation performance, we pass the sum of the encoded condition and the noisy latent representation through such a function:

$$y = |x| \cdot \left(\frac{2}{1+e^{-x}}\right) + \delta \tag{7}$$

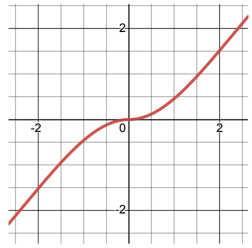

Figure 2. The graph of Eq. (7) when $\delta = 0$.

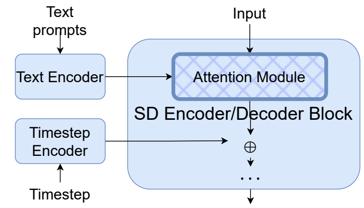

Figure 3. The structure of the encoder and decoder block in original diffusion model and ControlNet.

The δ is a randomized noise following distribution $\mathcal{N} \sim (0,1)$. The noise δ is randomized at the beginning of the fine-tuning and fixed during the fine-tuning. The server or the attacker has no access to δ. The function graph of Eq. (7) is shown in Fig. 2. The functionality of this function is to prevent the attacker from inferring the sum of the latent representation and encoder condition. To maintain image generation performance, we adopt a symmetric design with an SiLU-like shape. The SiLU function is a widely used activation function, which can help improve the performance of neural networks. As δ is fixed during the fine-tuning, the quality of the summation will not be degraded.

4.5. Prompt-Hiding Fine-Tuning

Apart from protecting image privacy, text prompts can also contain private information. In the original ControlNet, for each encoder and decoder block in the diffusion model and control network, text prompts are input into the blocks, and attention modules are applied to allow the generated images to learn the prompts. As a result, clients must upload text prompts or features to the server, exposing raw text information and risking privacy leaks. Uploading text encoder output features may prevent the server from accessing raw text, but the server can still use Carlini et al. 's method [2] to extract fine-tuning data using text features as inputs.

As a result, to hide prompts from the server, we propose not to send the text prompts to the server directly. During the fine-tuning, only the SD Encoder Block 1 in Fig. 1 on the clients, will take in text prompts as the input. The text prompts will not be uploaded to the server. Therefore, other encoder and decoder blocks in ControlNet, situated on the server, won't utilize text prompts as inputs. Removing text prompts will not affect the performance of condition and image encoders as they are irrelevant to prompts. However, the input distribution of server-side encoders and decoders has changed. To maintain high-quality image generation, we introduce the following prompts-hiding training methods.

As the diffusion model is frozen and able to always keep the generation performance of a well-trained diffusion model while the control network needs further training, we use different policies for the encoder and decoder blocks in the

Table 2. Comparison of image generation and privacy preservation among different deployments over Scribble condition

Methods	Performance		Privacy			
			CelebA		ImageNet	
	FID↓	CLIP↑	PSNR↓	SSIM ↓	PSNR ↓	SSIM ↓
Centralized	19.53	26.04	–	–	–	–
SL	19.46	26.87	14.41	0.37	8.17	0.35
Ours	13.45	26.85	13.15	0.37	7.34	0.47

control network and diffusion model. For blocks in control networks, no text features will be input and the attention modules will be replaced by self-attention modules for the condition features. Since we still need to further fine-tune the control network, the distribution drift caused by removal of text input will be mitigated during training.

To maintain the high image generation performance of the frozen diffusion model, we will keep the text attention modules but input a zero text feature. The zero text feature has the same feature dimension (by default 768) as the original but with a length of one and weights of zero. We need to keep the text input distribution consistent for these blocks since they will not be further trained. Otherwise, distribution drift will affect image generation performance.

5. Evaluation

5.1. Experimental Settings

We first briefly introduce experimental settings. More details are in Appendix D.

Execution Environment. We conduct all experiments on PLATO [19], an open-source research framework for deploying decentralized training on multiple devices. We can use PLATO to deploy the server and clients of large-scale decentralized training on separate devices conveniently.

Models and Datasets. For the pre-trained models, we used Stable Diffusion V-1.5 and ControlNet V-1.1. The autoencoder is from the pre-trained CLIP model with ViT-Large-Patch14 [30]. We use MS-COCO [21] as the training dataset for fine-tuning diffusion models to generate high-quality images with given conditions. It is a common dataset for fine-tuning diffusion models and text-to-image tasks.

Evaluation Metrics. For comparing performance, we must verify that the privacy-preserving method does not harm image generation and that an adversary cannot reconstruct private images. For the first objective, we use Fréchet Inception Distance [15] (FID) to evaluate generated image quality and the CLIP score [30] to assess whether prompts and generated images match (in range of $[0, 100]$). Lower FID indicates better image quality, while a higher CLIP score shows better alignment between text prompts and generated images. For the second objective, we use PSNR and SSIM@. More details are in Appendix C.3.1. Lower PSNR and SSIM indicate reconstructed images are less similar to private data, meaning better privacy-preserving effectiveness.

Conditions. Within the realm of conditional image generation, various tasks involve different conditions. We assess three types: canny lines, scribbles, and segmentation maps. These conditions range from detailed to coarse, with varying line detail. Examples appear in Appendix E.

Sampling. During the training process of ControlNet, the timestep is sampled among the range of $[t_s, t_{\max}]$. With the default k and β_0, we say $\epsilon_s = \epsilon(t_s, k, \beta_0)$. So, according to Eq. (6), during the training process, the privacy budget is equal to or larger than ϵ_s. Because we need to ensure that the privacy of every image is preserved, when we evaluate the effectiveness of privacy-preserving methods, in terms of both numerical data and visualization, we consider the worst case of least noise added and sample the timestep as t_s and send the intermediate features to the server.

5.2. Implementation

We briefly introduce the implementation of our methods and baselines. More details about the implementation are in Appendix F. For our privacy-preserving methods, we set the $t_{\max}$, k, and β_0 as default in ControlNet [45] which are 1000, 1.115×10^{-5} and 8.85×10^{-4} respectively. If t_s is too big, the sampling range will be too small to get enough samples. If t_s is too small, no privacy protection will be guaranteed. Hence, we set the t_s around middle point which is 536, which results in $\epsilon_s \approx 8$. The tuning of hyperparameter is in Appendix G.

We denote our structure without any privacy-preserving methods implemented (Sec. 3.2) as **Ours**. We implement our methods in three ways: only protecting conditions (**Ours+c**), only hiding prompts (**Ours+t**) and both (**Ours++**). The training latency remains the same after adding our privacy-preserving methods.

Implementation of baselines. We compare our methods with several state-of-the-art privacy-preserving methods which can be used for split learning with ControlNet and conventional training options.

LDP rr is a local differential private [8] mechanism with random response. The privacy budget is 2.

LDP number means the mechanism in Definition 4.1. We have three values for privacy budgets: 0.1, 0.3, and 0.5.

Add number means the mechanism of adding Gaussian noise on the raw data according to the distribution $\mathcal{N} \sim (0, \sigma^2)$. We have two numbers: $\sigma^2 = 1$ and $\sigma^2 = 50$.

Mixup is the method of mixing up data proposed in DataMix [23] and CutMix [28]. We mix four images together which is the same as the batch size.

PS is the method called patch shuffling [41, 43]. The patch size is set to 4, same as the batch size.

Table 3. Comparison of image generation and privacy preservation among different methods over Canny and Segmentation conditions

Condition	Canny						Segmentation				
	Performance		Privacy				Performance		Privacy		Attack works?
			CelebA		ImageNet				CelebA		
Methods	FID ↓	CLIP ↑	PSNR ↓	SSIM ↓	PSNR ↓	SSIM ↓	FID ↓	CLIP ↑	PSNR ↓	SSIM ↓	
Centralized	11.60	26.42	–	–	–	–	15.23	26.82	–	–	✓
SL	11.46	26.61	18.54	0.89	23.10	0.94	17.74	27.76	11.80	0.45	✓
Ours	18.59	26.21	18.86	0.73	22.84	0.86	14.35	26.92	12.18	0.49	✓
Ours+t	16.80	26.20					15.68	26.70			
Ours+c	14.52	26.80	17.45	0.51	21.74	0.70	15.05	26.85	1.68	0.46	×
Ours++	16.80	26.50					16.32	26.39			
LDP rr	18.11	27.22	18.86	0.82	23.77	0.97	17.49	27.23	14.92	0.72	✓
LDP 0.1	18.00	27.15	16.84	0.03	19.70	0.04	17.96	27.15	7.56	0.33	×
LDP 0.3	17.28	27.12	18.65	0.79	23.33	0.88	17.21	27.13	8.41	0.36	✓/×
LDP 0.5	12.27	26.53	19.81	0.90	24.31	0.95	17.46	27.21	11.21	0.51	✓
Add 1	11.77	26.60	25.69	0.98	31.02	0.995	17.51	27.30	22.96	0.88	✓
Add 50	19.69	26.84	25.53	0.99	30.60	0.99	17.60	27.29	23.05	0.90	✓
Mixup	401.62	13.54	17.96	0.14	22.84	0.19	384.24	13.99	13.45	0.73	×/✓
PS	17.39	27.16	21.25	0.95	25.85	0.98	17.62	27.22	22.64	0.92	✓

Arrow directions indicate superior image quality and increased difficulty in recognizing reconstructed data.
For privacy: ✓ and × whether the attack is able to reconstruct condition image. − means not applicable.

Centralized means images generated by the well-trained ControlNet. We directly use the downloaded models to generate images. This is a production-level baseline.

SL is the deployment of ControlNet with SL without any privacy-preserving methods applied (Sec. 3). We fine-tune ControlNet following steps of conventional SL.

5.3. Comparison Results

We present quantitative results in Tab. 3. The conclusion is that from numerical data and visualization, we can see that **our method can protect privacy without loss of image generation quality** tailored for ControlNet based diffusion models. The methods that can generate images correctly cannot preserve privacy as well. Methods that preserve privacy well cannot generate high-quality images. To improve image quality, a smaller disruption magnitude (e.g., lower privacy budget) is needed. Although advanced methods like Mixup and PS can protect privacy in some cases, they still fail to generate images of good quality that meet the conditions.

One of our new insights is that all previous privacy-preserving methods try to propose a general method for split learning, overlooking the variance between different use cases. We can easily extend methods like DataMix from image classification to different tasks. However, they cannot achieve satisfactory performance when we really verify them on the task of image generation. **Our privacy-preserving method is tailored for ControlNet based diffusion models**, considering the specialty of the overall model structure of diffusion models to how prompts are processed.

5.3.1 Maintenance of Image Generation Performance

To assess **image generation performance** with different privacy-preserving methods, we generate images under three conditions: canny, scribble, and segmentation. Examples are shown in Fig. 4 and Appendix E. We achieve image quality comparable to production-level ControlNet in centralized training. Apart from FedAvg, which does not work for ControlNet, methods like PS and previously proposed LDP mechanisms fail to generate images conforming to the conditions. Mixup cannot even generate a natural image.

An interesting result is that our designed split learning structure not only improves the efficiency but also improves the quality of generated images, reflected by FID. For example, on scribble conditions, we can improve FID from 19.53 to 13.45. This is possible as the pre-trained CLIP model is well-trained on large datasets. While for other methods, such as LDP Gaussian noise adding, though they can provide strong privacy protection with a small privacy budget, they need to sacrifice image generation quality. Furthermore, our methods preserve data privacy regardless of the number of samples on each client. Clients only need to make inferences from our designed structure. The results of inference are irrelevant to the number of samples passed through the models. Another reason is that our methods do not mix several training samples like what Mixup and Patch Shuffling did.

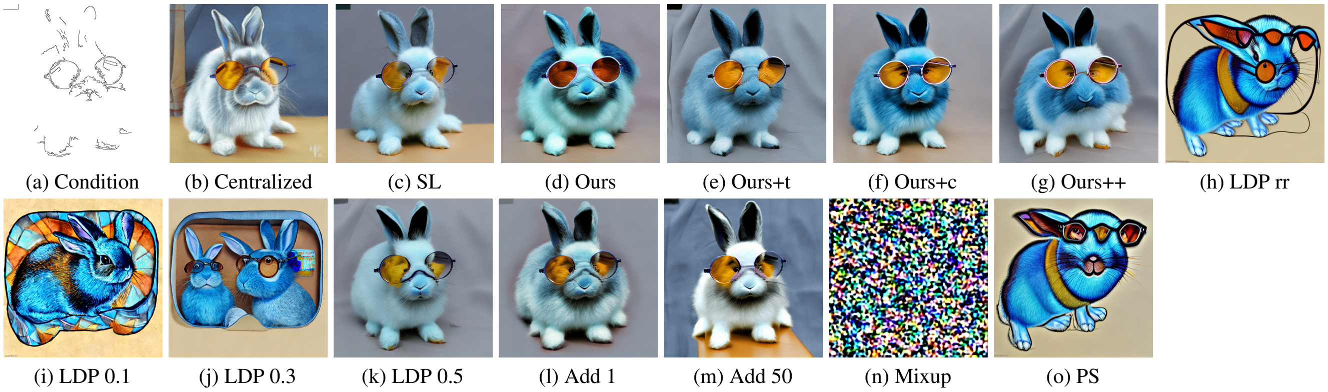

Figure 4. **Image generation:** Images of higher quality means better. Randomly selected and non-cherry-picked examples of images generated with the Canny condition under different methods. The text prompt is *a blue rabbit with glasses*.

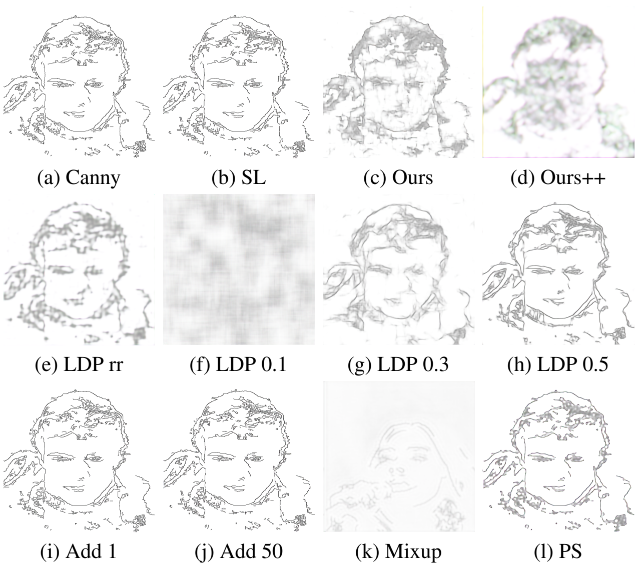

Figure 5. **Privacy preserving:** Higher distortion means better privacy preservation. Randomly selected and non-cherry-picked examples of reconstructed condition images of condition *canny*.

5.3.2 Privacy-preserving Ability

One observation is that depending on the granuality of different conditions, the needs of privacy-preservation is different. For segementation condition, when trivial split learning is implemented, the attacker is unable to reconstruct private images on ImageNet dataset where PSNR is 11.53 and SSIM is 0.50. When our method deployment structure is used, PSNR is 9.95 and SSIM is 0.47. For scribble condition, we can preserve the privacy by adopting our deployment method. The result is shown in Tab. 2.

While for Canny on both datasets and Segmentation on CelebA, a privacy-perserving solution is needed. For canny conditions, we can see results in Tab. 3 attackers find it easier to reconstruct private data, while for scribble conditions, it is much harder. However, as canny contains richer information, including complex lines, protecting such conditions is crucial. **We analyze privacy concerns for individual conditions and datasets separately.** In segmentation, attack success depends on the datasets. In cases where split learning with the original and our structure can defend against existing attacks, our methods enhance privacy. For other cases, we can reduce PSNR from 11.80 to 1.68, protecting privacy. To get a straightforward sense of **privacy-preserving performance**, we show examples of reconstruction by inverse network-based attacks in Fig. 5 using condition canny.

6. Concluding the Remarks

In this paper, we address the challenge of fine-tuning ControlNet models with locally distributed data across multiple users, focusing on feasibility and privacy. Considering that clients cannot afford high GPU memory requirements for on-device training, we turn to split learning to solve such a problem where we first improve the structure so that the server does not need to send gradients back to the clients, greatly improving efficiency. For existing threats in practical settings, we propose differential private timestep sampling, a noise-confounding activation function, and prompts-hiding fine-tuning, based on the built-in mechanisms in diffusion models with tunable privacy budgets. We show convincing results from a wide array of experiments that our method can provide stronger privacy protection without loss of image generation performance and train the models faster than its state-of-the-art alternatives in the literature.

Acknowledgment

We thank Professor Baochun Li, University of Toronto for valuable comments that greatly improved the manuscript.

References

[1] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, pages 679–698, 1986. 2

[2] Nicolas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramèr, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. In *Proceedings of the 32nd USENIX Security Symposium (USENIX Security)*, pages 5253–5270, 2023. 3, 5, 12

[3] Yaojian Chen and Qiben Yan. Privacy-preserving diffusion model using homomorphic encryption. *arXiv preprint arXiv:2403.05794*, 2024. 2

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009. 13

[5] Jinshuo Dong, Aaron Roth, and Weijie J Su. Gaussian differential privacy. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 84(1):3–37, 2022. 4

[6] Jinhao Duan, Fei Kong, Shiqi Wang, Xiaoshuang Shi, and Kaidi Xu. Are diffusion models vulnerable to membership inference attacks? In *Proceedings of the International Conference on Machine Learning (ICML)*, 2023. 3

[7] Cynthia Dwork. Differential privacy: A survey of results. In *Proceedings of the International conference on theory and applications of models of computation (TAMC)*, pages 1–19, 2008. 4

[8] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N Rothblum, and Salil Vadhan. On the complexity of differentially private data release: Efficient algorithms and hardness results. In *Proceedings of the forty-first annual ACM symposium on Theory of computing (STOC)*, pages 381–390, 2009. 6

[9] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Theoretical Computer Science (TCS)*, 9(3-4):211–407, 2014. 3, 4

[10] Ege Erdoğan, Alptekin Küpçü, and A. Ercüment Çiçek. UnSplit: Data-oblivious model inversion, model stealing, and label inference attacks against split learning. In *Proceedings of the 21st Workshop on Privacy in the Electronic Society (WPES)*, page 115–124, 2022. 1, 3, 12, 13

[11] Otkrist Gupta and Ramesh Raskar. Distributed Learning of Deep Neural Network over Multiple Agents. *Journal of Network and Computer Applications (JNCA)*, 116:1–8, 2018. 1

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 770–778, 2016. 3

[13] Zecheng He, Tianwei Zhang, and Ruby B Lee. Model inversion attacks against collaborative inference. In *Proceedings of the 35th Annual Computer Security Applications Conference (ACSAC)*, pages 148–162, 2019. 3, 4, 12

[14] Wu Hecong. ControlLoRA Version 2: A Lightweight Neural Network To Control Stable Diffusion Spatial Information Version 2, 9 2023. 2

[15] 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. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pages 6629–6640, 2017. 6

[16] Alain Hore and Djemel Ziou. Image quality metrics: PSNR vs. SSIM. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, pages 2366–2369, 2010. 13

[17] Gary Huang, Marwan Mattar, Honglak Lee, and Erik Learned-Miller. Learning to align from scratch. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pages 764–772, 2012. 12

[18] Lianghua Huang, Di Chen, Yu Liu, Yujun Shen, Deli Zhao, and Jingren Zhou. Composer: Creative and controllable image synthesis with composable conditions. *arXiv preprint arXiv:2302.09778*, 2023. 2

[19] Baochun Li, Ningxin Su, Chen Ying, and Fei Wang. Plato: An open-source research framework for production federated learning. In *Proceedings of the 2023 ACM Turing Award Celebration Conference (ACM TURC)*, pages 1–2, 2023. 6

[20] Oscar Li, Jiankai Sun, Xin Yang, Weihao Gao, Hongyi Zhang, Junyuan Xie, Virginia Smith, and Chong Wang. Label leakage and protection in two-party split learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. 12

[21] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 740–755, 2014. 6, 11, 13

[22] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3730–3738, 2015. 12, 13

[23] Zhijian Liu, Zhanghao Wu, Chuang Gan, Ligeng Zhu, and Song Han. DataMix: Efficient privacy-preserving edge-cloud inference. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 578–595, 2020. 1, 3, 4, 6, 12

[24] Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023. 2

[25] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of he 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1273–1282, 2017. 11

[26] Fatemehsadat Mireshghallah, Mohammadkazem Taram, Prakash Ramrakhyani, Ali Jalali, Dean Tullsen, and Hadi Esmaeilzadeh. Shredder: Learning noise distributions to protect inference privacy. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 3–18, 2020. 3

[27] Chong Mou, Xintao Wang, Liangbin Xie, Jian Zhang, Zhongang Qi, Ying Shan, and Xiaohu Qie. T2I-Adapter: Learning

adapters to dig out more controllable ability for text-to-image diffusion models. *arXiv preprint arXiv:2302.08453*, 2023. 2

[28] Seungeun Oh, Jihong Park, Sihun Baek, Hyelin Nam, Praneeth Vepakomma, Ramesh Raskar, Mehdi Bennis, and Seong-Lyun Kim. Differentially private cutmix for split learning with vision transformer. In *Proceedings of the First Workshop on Interpolation Regularizers and Beyond at NeurIPS*, 2022. 3, 6

[29] Dario Pasquini, Giuseppe Ateniese, and Massimo Bernaschi. Unleashing the tiger: Inference attacks on split learning. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 2113–2129, 2021. 3, 12, 13

[30] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 8748–8763, 2021. 6, 11

[31] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022. 11

[32] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 44(3):1623–1637, 2020. 2

[33] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2022. 2

[34] Abhishek Singh, Ayush Chopra, Ethan Garza, Emily Zhang, Praneeth Vepakomma, Vivek Sharma, and Ramesh Raskar. DISCO: Dynamic and invariant sensitive channel obfuscation for deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12125–12135, 2021. 3

[35] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 2256–2265, 2015. 1, 2

[36] Tom Titcombe, Adam J Hall, Pavlos Papadopoulos, and Daniele Romanini. Practical defences against model inversion attacks for split neural networks. In *Proceedings of the ICLR 2021 Workshop on Distributed and Private Machine Learning (DPML)*, 2021. 3

[37] Praneeth Vepakomma, Otkrist Gupta, Abhimanyu Dubey, and Ramesh Raskar. Reducing leakage in distributed deep learning for sensitive health data. In *Proceedings of the ICLR AI for Social Good Workshop*, volume 2, 2019. 3

[38] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018. 1

[39] Taihong Xiao, Yi-Hsuan Tsai, Kihyuk Sohn, Manmohan Chandraker, and Ming-Hsuan Yang. Adversarial learning of privacy-preserving and task-oriented representations. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 12434–12441, 2020. 1, 3

[40] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1395–1403, 2015. 2

[41] Hengyuan Xu, Liyao Xiang, Hangyu Ye, Dixi Yao, Pengzhi Chu, and Baochun Li. Shuffled transformer for privacy-preserving split learning. *arXiv preprint arXiv:2304.07735*, 2023. 3, 6

[42] Hengyuan Xu, Liyao Xiang, Hangyu Ye, Dixi Yao, Pengzhi Chu, and Baochun Li. Permutation equivariance of transformers and its applications. In *Proceedings of The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 3

[43] Dixi Yao, Liyao Xiang, Hengyuan Xu, Hangyu Ye, and Yingqi Chen. Privacy-preserving split learning via patch shuffling over transformers. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 638–647, 2022. 1, 3, 4, 6, 12

[44] Jiwen Yu, Yinhuai Wang, Chen Zhao, Bernard Ghanem, and Jian Zhang. FreeDoM: Training-free energy-guided conditional diffusion model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 23174–23184, 2023. 2

[45] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3836–3847, 2023. 1, 2, 6, 11, 14

[46] Yuheng Zhang, Ruoxi Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn Song. The secret revealer: Generative model-inversion attacks against deep neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 253–261, 2020. 3, 12

[47] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 633–641, 2017. 2