DPAF: Image Synthesis via Differentially Private Aggregation in Forward Phase

Chih-Hsun Lin, Chia-Yi Hsu, Chia-Mu Yu, *Senior Member, IEEE,*
Yang Cao, *Member, IEEE*, Chun-Ying Huang, *Senior Member, IEEE*

***Abstract*—Differentially private synthetic data is a promising alternative for sensitive data release. Many differentially private generative models have been proposed in the literature. Unfortunately, they all suffer from the low utility of the synthetic data, especially for high resolution images. Here, we propose DPAF, an effective differentially private generative model for high-dimensional image synthesis. Unlike previous private stochastic gradient descent-based methods, which add Gaussian noise in the backward phase during model training, DPAF adds differentially private feature aggregation in the forward phase, which brings advantages such as reducing information loss in gradient clipping and low sensitivity to aggregation. Since an inappropriate batch size has a negative impact on the utility of synthetic data, DPAF also addresses the problem of setting an appropriate batch size by proposing a novel training strategy that asymmetrically trains different parts of the discriminator. We extensively evaluate different methods on multiple image datasets (up to images of 128×128 resolution) to demonstrate the performance of DPAF.**

***Index Terms*—Differential Privacy, Synthetic Dataset.**

I. Introduction

The widespread use of IoT devices involves the collection of data and the development of corresponding applications, which are typically encrypted to ensure secure transmission to server databases. As technological advancements rely on various data for analysis and multiple downstream tasks, for example, deep neural networks such as computer vision, natural language processing, speech recognition, etc., require a large amount of high-quality data. Unfortunately, much valuable data is privacy sensitive, and direct use of the data becomes infeasible. Synthetic data has been proposed as a means to overcome the above difficulty. In particular, synthetic data from generative models can have the same statistical information as the original data, and thus leads to great data utility, which means that the analytical result on the synthetic data is similar to the one on the original data. Synthetic data sampled from the data distribution estimated by generative models is usually assumed to be decoupled from the original data and therefore implies privacy. However, recent studies reveal that the generative models still leak privacy information [1], [2], [3].

Differential Privacy (DP) [4] is the gold standard for privacy. Differentially private deep learning (DPDL), a variant of deep learning with the DP guarantee, can be used to provably preserve the privacy of DL models. For example, Abadi et al. [5] propose differentially private stochastic gradient descent (DPSGD) to train a DP classifier by adding the Gaussian noise to the clipped gradient. A majority of subsequent works apply DPSGD during the training of generative adversarial networks (GANs) to derive differentially private generative adversarial networks (DPGANs). Although DPSGD remains the most popular method for training a DPDL model, DPSGD has negative [6] effects on the model utility. Thus, the proper design of DPGANs is far from trivial.

In this work, we aim to design a DPGAN for image synthesis by relying on DPSGD. The primary goal of our work is to generate high-dimensional images in a DP manner so that the downstream classification task can have high classification accuracy. It has been widely known that the utility of DPSGD-based DPGANs is degraded due to two factors, the information loss in gradient clipping and DP noise. In this sense, our proposed techniques are designed to reduce both the information loss (by minimizing the size of the gradient vector) in gradient clipping and DP noise (by minimizing the global sensitivity).

Several efforts have been made to reduce the information loss of DPSGD. For example, GS-WGAN [7] minimizes the information loss by exploiting the 1-Lipschitz property of WGAN [8]. However, its generative power is also limited by the WGAN structure. DataLens [9] compresses the gradient vector by keeping only the top-k values and then quantizing them. Since the non-top-k values in the gradient vector do not involve gradient clipping, more meaningful quantities can be retained from the top-k values. However, the success of DataLens relies heavily on training a large number of teacher classifiers, which hampers its practicality due to large amounts of GPU memory.

Our method takes a different approach, i.e., we perform a DP feature aggregation in the forward phase during the training of the NN. The postprocessing of DP ensures that the number of dimensions of the gradient vector can be effectively reduced. This greatly reduces information loss from gradient clipping in DPSGD. On the other hand, choosing a large batch size is a straightforward way to reduce the negative impact of DP noise, because the DP noise can be easier to cancel each other out. Unfortunately, a large batch size leads to much fewer updates of the discriminator, and thus the training cannot converge given a fixed number of epochs. In addition, due to the feature aggregation in our proposed method, the large batch size also easily makes features indistinguishable, which is detrimental to the utility of the synthetic data and, in turn, favors a small batch size instead. We address this dilemma through an elaborate design of the training strategy.

Contribution. The contributions are summarized below.

- We propose DPAF (**D**ifferentially **P**rivate **A**ggregation in **F**orward phase), an effective generative model for differentially private image synthesis. DPAF supports the conditional generation of high-dimensional images.
- We propose a novel framework to enforce DP during GAN training. In particular, we propose to place a differentially private feature aggregation (DPAGG) in the forward phase.

Chih-Hsun Lin, Chia-Yi Hsu, Chia-Mu Yu (corresponding author, chiamuyu@nycu.edu.tw), and Chun-Ying Huang are with National Yang Ming Chiao Tung University, Taiwan. Yang Cao is with the Tokyo Institute of Technology, Japan.

Together with a simplified instance normalization (SIN), DPAGG can not only have a natural and low global sensitivity, but also significantly reduce the dimensionality of the gradient vector. We also have a novel design of asymmetric model training, which solves the dilemma that a small batch cannot effectively reduce the DP noise, but a large batch will make features indistinguishable.
- We formally prove the privacy guarantee of DPAF. Furthermore, we conduct extensive experiments on DPAF on popular image datasets, CelebA (rescaled to 64×64) and FFHQ (128×128). Our experimental results show that DPAF outperforms the previous solutions on CelebA and FFHQ in terms of both classification accuracy (e.g., $[(0.802 - 0.700)/0.700] \times 100\% = 14.57\%$ improvement on CelebA-Gender with privacy budget $\varepsilon = 1$ compared to SOTA) and visual quality, as DPAF is characterized by its unique property that the generative capability is maximized for larger images.

II. Related Work

This paper primarily studies how to synthesize images with the DP guarantee. The works for DP tabular data synthesis [10], [11], [12] are not in our consideration. In addition, though DP generative models and DP classifiers [13], [14], [15], [16], [17] share DPSGD-based training strategies, the design of DP classifiers is beyond the scope of this paper.

Differentially Private Generative Models. Differentially private stochastic gradient descent (DPSGD) [5] and private aggregation of teacher ensembles (PATE) [18], [19] are two common techniques that achieve DP generative models. DPSGD trains a model by injecting DP noise into the gradient in each training iteration; i.e., the gradient is first clipped to ensure a controllable global sensitivity and then perturbed by a proper Gaussian noise. Nevertheless, gradient clipping in DPSGD incurs significant information loss, which severely compromises model accuracy. To improve the utility of DPDL models, PATE partitions the sensitive dataset into disjoint subsets and trains a teacher model for each subset. The noisy aggregated teachers vote to determine the class for the unlabeled public data. PATE possesses the advantage of no information loss, in contrast to DPSGD. Nonetheless, despite no information loss, the application of PATE to generative models is by no means trivial, because the generator may output synthetic samples with similar labels, and consequently the teacher models cannot learn to be updated effectively [20].

DPSGD-Based Approach. Although gradient clipping is beneficial in reducing global sensitivity, it leads to a dramatic loss of information and therefore hurts the model utility. Therefore, the majority of works focus on improving the DPSGD. Zhang et al. [21] cluster the parameters with similar clipping bounds, and add DP noise to different parts of the gradient. For better control of the noise scale, McMahan et al. [22] and Thakkar et al. [23] compute the clipping bound via adaptive clipping. GANobfuscator [24], GS-WGAN [7], and Xie et al.'s method [25], all based on the improved WGAN framework [8], reduce the noise scale by exploiting the Lipschitz property. In addition, GS-WGAN argues that the discriminator does not need to be trained via DPSGD because only the generator is released. By projecting the gradients onto a predefined subspace [26], [27], one can reduce the global sensitivity to lower the DP noise scale.

PATE-Based Approach. Through the teacher ensemble, PATE-GAN uses the noisy labels on the samples generated by the generator to train both the generator and the discriminator based on the PATE framework. However, an inherent assumption behind PATE-GAN is that the generator is capable of generating samples from the entire real sample space, which is not always true. G-PATE [28] achieves DP on the aggregated gradient from the teacher ensemble when backpropagating to update the generator, through the Confident-GNMax aggregator [19]. Similar to G-PATE [28], DataLens [9] enforces DP on the aggregated gradient from the teacher ensemble, but the aggregated gradient is compressed and quantized to limit the information loss of gradient clipping.

Non-Adversarial Learning-based Approach. DP-MERF [29] trains the generator by considering the maximum mean discrepancy (MMD) over random feature representations of kernel mean embeddings for both the data and generator distributions. Given the use of MMD, DP-MEPF [30], DP-NTK [31], and DP-HP [32] consider the pre-trained perceptual features, the features of the neural tangent kernels, and the Hermite polynomial features, respectively. Following a framework similar to DP-MERF, DP-Sinkhorn [33] and PEARL [34] train the generator instead by minimizing the Sinkhorn divergence with semi-debiased Sinkhorn loss and by minimizing the characteristic function distance, respectively. P3GM [35] considers a two-phase training for the variational autoencoder (VAE). In particular, P3GM trains the encoder and then trains the decoder with the frozen encoder. Such a two-phase training increases the robustness against noise for DP. Similar to P3GM, DP^2-VAE [36] and DPD-fVAE [37] are also VAE-based approaches. DPGEN [38] takes a different approach; it applies randomized response (RR) [4] to the movement direction of the Markov Chain Monte Carlo, avoiding the information loss of gradient clipping in DPSGD. Unfortunately, it only supports unconditional generation, so labeling the synthetic data consumes additional privacy. Recently, DPDC [39] has been proposed to synthesize data privately by taking advantage of dataset condensation [40]. The simplest form of DPDC is to apply Gaussian noise to the aggregated gradient, which is the sum of the gradients of random samples from a given class. Due to not only the sequential denoising behavior, but also the non-adversarial learning, DPDM [41], built upon the diffusion model [42], [43], is more robust to the DP noise. Ghalebikesabi et al's method [44] also relies on diffusion models and shows remarkably high test accuracies. However, their batch sizes are set up to 16,384, which is unacceptable for most commercial machines. DP-promise [45] also uses diffusion models, and they proposed a two-phase training process with DPSGD and SGD. PrivImage [46] carefully selected 1% of the public data by spending a small privacy budget to train the pre-trained model. Unlike training a generative model, DPSDA [47] obtains images by querying public APIs.

III. Preliminaries

Here, we introduce some necessary technical background for DPAF. First, we formulate DP. Then, we briefly describe generative models and transfer learning. Afterward, we will introduce how to train a neural network in a DP manner.

Differential Privacy. (ε, δ)-differential privacy [4], (ε, δ)-DP, is the *de facto* standard for data privacy. The privacy of (ε, δ)-DP comes from limiting the contribution of an input sample to the output distribution. More specifically, $\varepsilon > 0$ bounds the log-likelihood ratio of any given output when the

algorithm is run on two datasets that differ in one sample, while $\delta \in [0, 1]$ is the probability that certain outputs violate the above bound.

Definition 1: An algorithm $\mathcal{M}$ is (ε, δ)-DP if for all $\mathcal{S} \subseteq \text{Range}(\mathcal{M})$ and for any neighboring datasets $\mathcal{D}$ and $\mathcal{D}'$,

$$\Pr[\mathcal{M}(\mathcal{D}) \in \mathcal{S}] \leq e^{\varepsilon} \Pr[\mathcal{M}(\mathcal{D}') \in \mathcal{S}] + \delta. \quad (1)$$

Throughout this paper, $\mathcal{D}$ and $\mathcal{D}'$ are neighbors if $\mathcal{D}$ can be obtained by adding or removing a sample from $\mathcal{D}'$ in the unbounded DP sense [48]. DP can quantify privacy loss; i.e., ε in Eq. (1), called the *privacy budget*, measures the *privacy loss*. In essence, privacy gets worse with an accumulated ε. DP can be achieved by applying the Gaussian mechanism G_σ, where the zero-mean Gaussian noise with appropriate variance is added to the algorithm output. More specifically, given a function f, $G_\sigma \circ f(x) \triangleq f(x) + N(0, \sigma^2)$ satisfies (ε, δ)-DP for all $\varepsilon < 1$ and $\sigma > \sqrt{2\ln(1.25/\delta)}\Delta_{2,f}/\varepsilon$, where the ℓ_2-sensitivity of f is defined as $\Delta_{2,f} \triangleq \max_{\mathcal{D},\mathcal{D}'} ||f(\mathcal{D}) - f(\mathcal{D}')||_2$ for neighboring $\mathcal{D}$ and $\mathcal{D}'$.

DP has the following useful characteristics. First, repeated access to sensitive data leads to an accumulation of privacy loss. Such privacy loss can be bounded by the sequential composition theorem or higher-order techniques (e.g., moments accountant [49], [5]). Second, DP is not affected by postprocessing. Formally, $g \circ \mathcal{M}$ for any data-independent mapping g still satisfies (ε, δ)-DP, given that $\mathcal{M}$ is (ε, δ)-DP.

Generative Adversarial Network (GAN). A GAN consists of two components, a generator G and a discriminator D. Specifically, G learns to output synthetic samples, while D, which takes as input both the synthetic samples and the sensitive dataset, is trained to discriminate between real and fake samples. Formally, given the real sample x and a sampled noise z, D is trained with the loss function $\mathcal{L}_D = -\log D(x) - \log(1 - D(G(z)))$. On the other hand, G is trained with the loss function $\mathcal{L}_G = -\log D(G(z))$.

GAN can be extended to conditional GAN (cGAN) [50]. The difference between GAN and cGAN is that the latter can generate samples with a specific class (aka conditional generation), while the former cannot. cGAN can also be instantiated by the generator-discriminator architecture above. In addition, the class label is usually added to either the generator or the discriminator (or both) to ensure that the synthetic sample is consistent with the input class label. Examples of cGANs are AC-GAN [51] and cDCGAN [52].

Transfer Learning. Usually, a DNN model consists of two parts, feature extractor (FE) and label predictor. Transfer learning aims to effortlessly train a target model for a specific task by leveraging the pre-trained FE for another but related task. In particular, the target model is composed of the pre-trained FE and a randomly initialized label predictor. During the training of the target model, the FE is frozen and only the label predictor is updated with the training set from the target task. Formally, let $M(x)$ be the function of the target model with x as the input vector. M can be represented as $M(x) = M_c(M_f(x, \theta_f), \theta_c)$, where M_f, M_c, θ_f, and θ_c denote the FE, the label predictor, the parameters of the FE, and the parameters of the label predictor, respectively. A typical approach to transfer learning is to optimize the objective, $\min_{\theta_c} \sum_{x \in X, y \in Y} L(M_c(M_f(x, \theta_{f0}), \theta_c), y)$, where θ_{f0} denotes the parameters of the pre-trained FE, L is a loss function, and X and Y are the training data and labels, respectively.

Differentially Private Stochastic Gradient Descent. Differentially private stochastic gradient descent (DPSGD) is the most popular technique to train a DPDL model. Given a training set $\{(x_i, y_i)\}_{i=1}^N$, the update of the ordinary stochastic gradient descent (SGD) is formulated as $w^{t+1} = w^{(t)} - \eta_t \frac{1}{B} \sum_{i \in \mathcal{B}_t} \nabla \mathcal{L}(w^{(t)}, x_i, y_i)$, where $\mathcal{L}(w, x, y)$ is the loss function with the model parameter w, input sample x, and label y, and $\mathcal{B}_t$ is the set of samples at iteration t with $|\mathcal{B}_t| = B$. As the gradient has an unbounded sensitivity, one has to clip the gradient to ensure a bounded DP noise magnitude. Formally, the update of DPSGD can be formulated below.

$$w^{t+1} = w^{(t)} - \eta_t \left\{ \frac{1}{B} \sum_{i \in \mathcal{B}_t} \text{clip}_u \left(\nabla \mathcal{L}(w^{(t)}, x_i, y_i) \right) + \frac{\sigma u}{B} \xi \right\}, \quad (2)$$

where η_t is the learning rate, ξ is sampled from the zero-mean Gaussian distribution, σ specifies the standard deviation of the added noise, and clip_u is defined as $\text{clip}_u(v) = \min\{1, \frac{u}{||v||_2}\} \cdot v$ with u as a manually configurated clipping threshold. Usually, u is used as the ℓ_2-sensitivity for SGD.

IV. Threat Model

In practice, DL models are usually trained with privacy-sensitive data. Thus, keeping the training data private is a major concern before the real-world deployment of DL-based systems. Unfortunately, given access to the classifiers only, the attacker can still infer whether a specific sample is in training set by launching a membership inference attack (MIA) [53], [54]. It has been proven that MIA can apply to not only classifiers [53] but also GANs [1], [2], [3]. Furthermore, compared to MIAs, model inversion attacks [55] aim to recover the training samples. Though certain defenses have been developed, the reconstruction attack can still work [56].

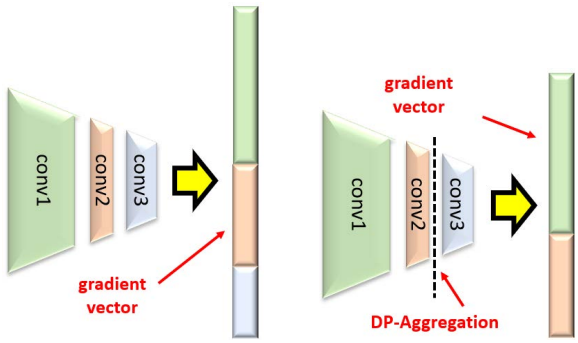
Goal. DP can protect against MIAs and training data reconstruction. Thus, our goal is to ensure the DP guarantee of the GANs while maintaining a high utility. More specifically, we assume that only the generator of a GAN is released for data synthesis. In other words, the attacker has no access to the discriminator. An attacker's access to the generator should not lead to privacy leakage of training samples, even if the attacker launches MIAs and the data recovery attacks mentioned above. In particular, maintaining high utility while ensuring privacy in DPGANs is very challenging because the DP noise can dramatically hinder the training of DPGANs. Here, the utility in our consideration includes the predicting accuracy of the classifier trained on synthetic samples and tested on real samples, and the visual quality of the synthetic samples (see Section VI-A).

V. Proposed Method

We give an overview in Section V-A, present DPAF in Section V-B, and have discussions of the design rationale behind DPAF in Section V-C.

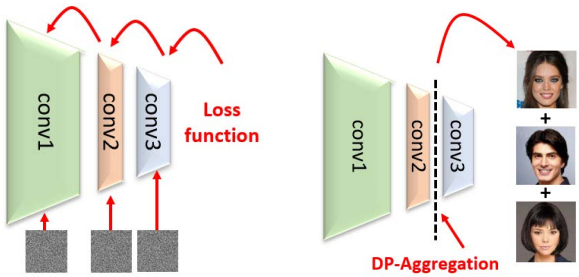
A. Overview

Unlike previous DPGANs, DPAF has a fundamentally different design where DP feature aggregation is performed in the forward phase. DPAGG, the procedure for implementing DP feature aggregation, not only sums the vectors from the forward layer but also requires that each vector undergo simplified instance normalization to ensure each feature vector is bounded (detail in **The Design of DPAGG** in Section V-B). The aggregated feature makes the image features more robust against the DP noise. On the other hand, the DP feature



(a) w/o DP aggregation. (b) w/ DP aggregation.

Fig. 1: The illustration of the impact of DP feature aggregation on the size of the gradient vector.



(a) w/o DP aggregation. (b) w/ DP aggregation.

Fig. 2: The impact of DP feature aggregation on the gradient structure preservation during the backpropagation.

aggregation in the forward phase implies a shortened gradient vector, resulting in a significant reduction of information loss in gradient clipping. DPAF is also characterized by the use of a simplified instance normalization that preserves fine-grained features and reduces ℓ_2 sensitivity. Overall, the five advantages of performing DP feature aggregation in the forward phase are summarized below.

Reduction of Information Loss in Gradient Clipping. The first advantage is the dimensionality reduction of the gradient vector, as illustrated in Figure 1. More specifically, gradient clipping in DPSGD inevitably leads to information loss. However, gradient clipping has less impact on shorter gradient vectors, resulting in less information loss. As shown in Figure 1a, the gradient vector to be clipped will be longer if DP feature aggregation is not used. On the contrary, as shown in Figure 1b, conv3 has been privatized after DP feature aggregation due to the postprocessing property of DP, and can be updated by SGD. As a result, since only conv1 and conv2 need to be updated by DPSGD, the information loss due to gradient clipping can be mitigated.

Better Preserving Gradient Structure. During backpropagation, DPSGD applies noise to the weights in a layer-by-layer manner, as shown in Figure 2a, which makes training more difficult because such an updating process destroys the inherent structure of the gradient vector. Thus, the second advantage is to better preserve the gradient structure. This is due to the fact that the aggregated DP vector is a vector of aggregated noisy image features. As shown in Figure 2b, since the aggregated features still have the inherent semantics, the corresponding noisy version remains meaningful. On the other hand, as also shown in Figure 2b, conv3 can be updated by

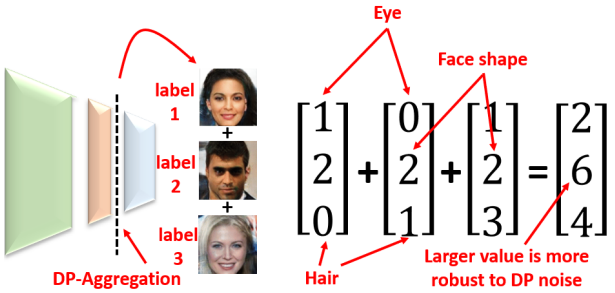


Fig. 3: The illustration of the better robustness against the DP noise after the feature aggregation.

SGD instead of DPSGD, which better preserves the inherent gradient structure. In this case, only layers (e.g. conv2) need to be updated by DPSGD, and as a result, only a small fraction of the parameter structure will be affected by DPSGD.

Better Robustness against DP Noise. The aggregation of the features from samples makes the aggregated feature more robust to the DP noise, because the DP noise is added after the feature value summation. This is illustrated in Figure 3, where each individual feature is relatively susceptible to the DP noise, but the aggregated one has the larger values and, as a result, better robustness.

Better Preserving Image Features. From Figure 3, we know that the aggregated feature vector, which is robust to the DP noise, helps in synthesizing realistic samples (because it is backpropagated to update G), but such synthetic samples may be irrelevant for a particular label. In fact, the feature values should have similar numerical ranges, otherwise D will only pay attention to the features with large values and ignore the features with small values. As a consequence, G cannot be updated well. We propose to use a simplified instance normalization (SIN) to ensure that fine-grained features can be learned. Specifically, SIN is applied to each feature map individually. Then, the feature vector (concatenated normalized feature maps) undergoes aggregation. For example, this helps in synthesizing faces with consistent gender in the conditional generation of faces. In other words, in general, without SIN, due to the imbalance of feature values, some feature values will be devoured by the others, resulting in the disappearance of certain important feature values that are related to the specific class.

Low Global Sensitivity. The fifth advantage is the low ℓ_2-sensitivity of the SIN-and-aggregation operation. More specifically, as mentioned above, the feature maps need to be normalized and then concatenated as the feature vector before the aggregation. We find that the ℓ_2-sensitivity of such an SIN-and-aggregation operation can be calculated as a relatively small and controllable value $\sqrt{mp}$, where m is the number of feature maps and the size of the feature map is $p \times p$. Properly setting $\sqrt{mp}$ effectively reduces the noise magnitude, thereby raising the utility (see Section V-C).

B. DPAF

Here, we present DPAF (**D**ifferentially **P**rivate **A**ggregation in **F**orward phase), an effective generative model for differentially private image synthesis. The architecture of DPAF is illustrated in Figure 4. The notation table summarizing the frequently used notations can be found in Table XVIII in Supplementary Materials. DPAF is trained by using transfer

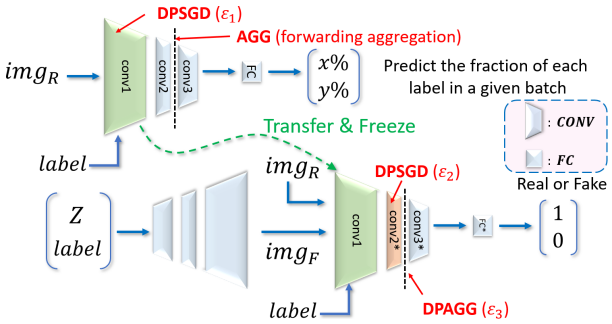


Fig. 4: The model architecture of DPAF.

learning. Hence, similar to transfer learning, DPAF has two phases, training a classifier first and training a GAN, both in the DP manner. We describe below each of them below.

Although DPAF uses transfer learning as a subroutine to improve utility, it does not use warm start [21], a technique to improve utility by exploiting the extra data with similar data distribution, because it is not common to find such data for arbitrary sensitive datasets.

1) Training a Classifier Before Transfer Learning: The workflow of training the classifier C in DPAF before transfer learning is shown in Figure 5. The classifier C is identical to the discriminator of cDCGAN (in fact, a standard convolutional neural network (CNN), see Section V-C), where there are three parts of convolutional layers (conv1, conv2, and conv3) and some fully connected (FC) layers, except that an aggregation layer is added between conv2 and conv3. The actual numbers of layers for the convolutional and FC layers, and the number of neurons in the input layer, depend on the size of the input image. We follow the common setting that the height/width of the feature maps in the next convolutional layer is half that of the current convolutional layer.

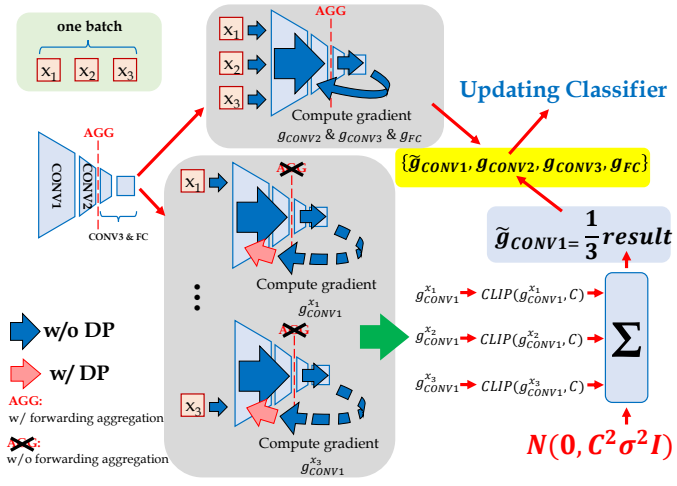


Fig. 5: The procedure of updating classifier.

The ordinary CNN is designed to classify inputs. However, because we introduce an aggregation layer (AGG) between conv2 and conv3, which aggregates multiple normalized features in a batch, obviously the CNN can no longer output the predicted class for a given input image and label. Instead, C here is designed to predict the percentage of each class in a given batch, as shown in Figure 4. To achieve the above goal, C is trained with the labeled sensitive data (img_R in

Algorithm 1: Training of DPAF

1 **Notation**: number of batches $\mathbb{B}$, mean square error loss function $\mathcal{L}_{\text{MSE}}$, binary cross-entropy loss functions $\mathcal{L}_{\text{BCE}}$, $\mathcal{L}'_{\text{BCE}}$, $\mathcal{L}''_{\text{BCE}}$, asymmetry multiplier μ, number of critic iterations per generator iteration n_{critic}
/* for-loop below trains the classifier C */
2 **for** $i = 1$ *to* $\mathbb{B}$ **do**
3 compute $\mathcal{L}_{\text{MSE}}$ over the i-th batch
4 perform SGD for updating conv2, conv3, and FC and then perform DPSGD(ε_1) for updating conv1 in one round of backpropagation
5 conv1*$\leftarrow$ conv1 // conv1* and conv1 share parameters
6 **for** $i = 1$ *to* $\mathbb{B}$ **do**
/* computing $\mathcal{L}_{\text{BCE}}$ on D with DPAGG(ε_3), $\mathcal{L}'_{\text{BCE}}$ on D that replaces DPAGG(ε_3) by AGG, and $\mathcal{L}''_{\text{BCE}}$ on D without DPAGG(ε_3), respectively */
7 compute $\mathcal{L}_{\text{BCE}}$ over the i-th batch
/* code below asymmetrically trains D */
8 **if** $i\%\mu = 0$ **then**
9 compute $\mathcal{L}'_{\text{BCE}}$ over the $[i-\mu+1, i]$-th batches
10 SGD for updating conv3* and FC* by $\mathcal{L}_{\text{BCE}}$
11 DPSGD(ε_2) for updating conv2* by $\mathcal{L}'_{\text{BCE}}$
12 **else**
13 SGD for updating conv3* and FC* by $\mathcal{L}_{\text{BCE}}$
/* the code below trains G */
14 **if** $i\%n_{critic} = 0$ **then**
15 compute $\mathcal{L}''_{\text{BCE}}$ over each sample from the i-th batch
16 SGD for update of G

Figure 4) through the mean square error (MSE) loss function, $\mathcal{L}_{\text{MSE}}$. Afterward, in one round of the backpropagation, SGD is applied to conv2, conv3, and FC for the update of the corresponding parameters, while the DPSGD with the privacy budget ε_1, DPSGD(ε_1), is applied to conv1, because only conv1 will be recycled to be used after transfer learning. Lines 2~4 in Algorithm 1 correspond to the above training procedures. After training, conv2, conv3, and FC are discarded and will not be released.

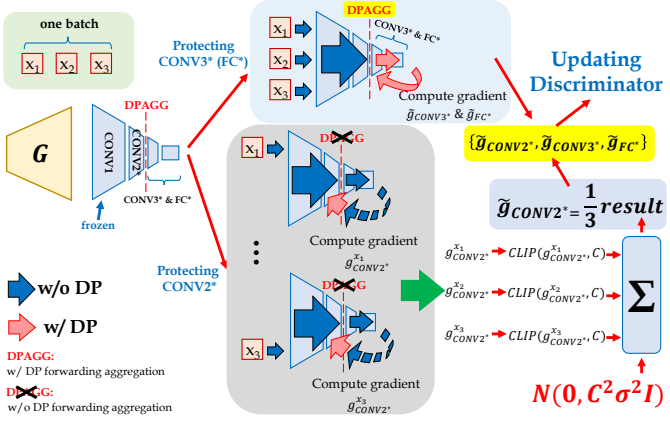


Fig. 6: The procedure of updating GAN.

2) Training a DPGAN After Transfer Learning: In this phase, we aim to train a DPGAN such that a generator satisfying DP can be released. GAN is known to be composed of two parts, a generator G and a discriminator D. In DPAF, the architecture of D is identical to C; the workflow of training D in DPAF is shown in Figure 6. The conv1 in C is transferred to be the conv1 in D; i.e., C and D share the same conv1 (Line 5 in Algorithm 1). Such a conv1 does not leak privacy as it is trained by DPSGD, though conv2, conv3, and FC are trained by SGD[1] (see the formal proof in Theorem 1 and Corollary 1 in Section VII-C). Here, conv2*, conv3*, and FC* are randomly initialized. Unlike C, where an AGG is placed between conv2 and conv3, a DP feature aggregation layer with the privacy budget ε_3, DPAGG(ε_3), is placed between conv2* and conv3* of D. The architecture of G is a reverse of D without DPAGG(ε_3).

Training the DPGAN in DPAF is similar to training an ordinary GAN; i.e., we iteratively train D first and then the G until the convergence. D takes as input the sensitive images (img_R in Figure 4), synthetic images (img_F in Figure 4), and the label. Given that conv1 is frozen during the training due to transfer learning, D is trained to differentiate between real and synthetic images. The general guideline of training D is that after binary cross entropy (BCE) loss function $\mathcal{L}_{\text{BCE}}$ is calculated on D with DPAGG(ε_3), conv3* and FC* can be updated via SGD because such an update of conv3* and FC* still satisfies the DP according to the postprocessing of DPAGG(ε_3). Consider $\tilde{D}$ as the discriminator D that replaces DPAGG(ε_3) by AGG. In addition, another BCE loss function $\mathcal{L}'_{\text{BCE}}$ is calculated on $\tilde{D}$. conv2* will be updated by using DPSGD(ε_2) based on $\mathcal{L}'_{\text{BCE}}$. Lines 7~13 in Algorithm 1 correspond to the training of D. The details about $\mathcal{L}_{\text{BCE}}$ and $\mathcal{L}'_{\text{BCE}}$ are related to our proposed asymmetric training and will be described later.

Consider $\hat{D}$ as the discriminator D without DPAGG(ε_3); i.e., $\hat{D}$ can be seen as a standard CNN. After updating D n_{critic} times, the BCE loss function $\mathcal{L}''_{\text{BCE}}$ is calculated on $\hat{D}$ and is backpropagated to update G through SGD. n_{critic} is the number of critic iterations per generator iteration for the better training [8]. The design of skipping the aggregation in D when training G can be attributed to the fact that we aim to learn how to generate a single image, instead of a mix of images. Note that $\mathcal{L}_{\text{BCE}}$, $\mathcal{L}'_{\text{BCE}}$, and $\mathcal{L}''_{\text{BCE}}$ all work on the same D, but depending on which part of D needs to be updated, different components of D are ignored. Lines 14~16 in Algorithm 1 correspond to the training of G.

The Design of DPAGG. AGG can be implemented via two steps. First, the normalized feature maps are concatenated as a feature vector. Second, the feature vectors from different samples in a batch are aggregated. We can have DPAGG when we apply the Gaussian mechanism to the aggregated feature vector derived from AGG. Below we describe how our instance normalization works.

Inspired by [57], [58], we propose to use a simplified instance normalization (SIN) to not only ensure the balance of feature values but also, more importantly, derive a bound of the global sensitivity of AGG. SIN can be formulated as follows.

[1]GS-WGAN (Lines 17 and 19 in Algorithm 1 of [7]), G-PATE (Lines 10 and 14 in Algorithm 1 of [28]), and DataLens (Lines 10 and 14 in Algorithm 1 of [9]) have a similar design, where some parts of the model are trained by SGD but eventually discarded while the remaining parts are trained by DPSGD for the eventual release.

batch #	batch 1	batch 2	batch 3	batch 4	batch 5	batch 6	batch 7	batch 8	batch 9	batch 10
update G?			G			G			G	
update D?	conv3* FC*	conv3* FC*	conv3* FC*	conv3* FC*	conv3* FC*	conv3* FC*	conv3* FC*	conv3* FC* conv2*	conv3* FC*	conv3* FC*

Fig. 7: The asymmetric training with $\mu = 8$ and $n_{\text{critic}} = 3$.

$$\begin{aligned}
\mu_{i_1 i_2} &= \frac{1}{HW}\sum_{i_3=1}^{H}\sum_{i_4=1}^{W} x_{i_1 i_2 i_3 i_4}, \\
\sigma^2_{i_1 i_2} &= \frac{1}{HW}\sum_{i_3=1}^{H}\sum_{i_4=1}^{W}\left(x_{i_1 i_2 i_3 i_4} - \mu_{i_1 i_2}\right)^2, \\
\widehat{x_{i_1 i_2 i_3 i_4}} &= \frac{x_{i_1 i_2 i_3 i_4} - \mu_{i_1 i_2}}{\sqrt{\sigma^2_{i_1 i_2}}},
\end{aligned} \tag{3}$$

where $\mu_{i_1 i_2}$ is the mean of feature map $X_{i_1 i_2}$, $\sigma^2_{i_1 i_2}$ is the variance of $X_{i_1 i_2}$, i_1 is the index of the image in the batch, i_2 is the feature channel (color channel if the input is an RGB image), H is the height of the feature map, W is the width of the feature map, $x_{i_1 i_2 i_3 i_4} \in \mathbb{R}$ is an element of feature map $X_{i_1 i_2}$, $\widehat{x_{i_1 i_2 i_3 i_4}}$ is the new value of $x_{i_1 i_2 i_3 i_4}$ after SIN. One can easily see that SIN is different from ordinary instance normalization (IN) in that SIN does not have learnable parameters, center and scale [57], [58]. Concretely, each normalized feature map (through SIN) is guaranteed to have the same ℓ_2-norm p, where $p \times p$ is the size of feature map.

Before applying the Gaussian mechanism to the AGG output, we calculate the ℓ_2-sensitivity $\Delta_{2,\text{AGG}}$ of the AGG below:

$$\begin{aligned}
\Delta_{2,\text{AGG}} &= \sqrt{\sum_{x\in X_{11}}\left(\frac{x-\mu_{11}}{\sigma_{11}}\right)^2 + \cdots + \sum_{x\in X_{1m}}\left(\frac{x-\mu_{1m}}{\sigma_{1m}}\right)^2} \\
&= \sqrt{\frac{1}{\sigma^2_{11}}\sum_{x\in X_{11}}(x-\mu_{11})^2 + \cdots + \frac{1}{\sigma^2_{1m}}\sum_{x\in X_{1m}}(x-\mu_{1m})^2} \\
&= \sqrt{\sum_{j=1}^{m}\left[\frac{\|X_{1j}\|}{\sum_{x\in X_{1j}}(x-\mu_{1j})^2}\sum_{x\in X_{1j}}(x-\mu_{1j})^2\right]} \\
&= \sqrt{\|X_{11}\| + \cdots + \|X_{1m}\|} = \sqrt{p^2 + \cdots + p^2} = \sqrt{m}p,
\end{aligned} \tag{4}$$

where $||X||$ is the size of feature map X, m is the number of feature maps, x is an element of feature map X_{1j} for $j = 1, 2, \cdots, m$. In the above calculation of $\Delta_{2,\text{AGG}}$, we consider batch size 1 because we aim to know the amount of difference to which a single sample in the batch contributes.

Depending on the tasks, the normalization can be placed in a different position or even multiple layers [59], [60] for better training. We find that in addition to offering a fine-grained control of features similar to computer vision tasks, SIN in DPAF plays a unique role in bounding ℓ_2-sensitivity, though SIN is an easy modification of IN. Note that we apply SIN to only those feature maps just before DPAGG. Such a design is supported by our experiments that applying SIN in all the layers before DPAGG, in turn, degrades the utility because, unlike IN, SIN lacks the learnable parameters.

Asymmetric Training of D in DPAF. In fact, AGG asks for a smaller batch size because, otherwise, the features will be mixed and cannot be recognized. Nevertheless, a smaller

batch size, in turn, is harmful to DPSGD because the DP noise will make a greater impact on the gradient. Hence, DPAF prefers a larger batch size from the DPSGD point of view. We propose an asymmetric training strategy to resolve the contradicting requirements of setting a proper batch size. In essence, in the asymmetric training, when training D, we update conv3* and FC* through SGD for every iteration but update conv2* through DPSGD(ε_2) for every μ iterations, as shown in Figure 7. μ is called *asymmetry multiplier* because it determines the ratio of the privacy budget for conv2* and the budget for both conv3* and FC*.

More specifically, for each batch, $\mathcal{L}_{\text{BCE}}$ is calculated by feeding the samples to D and then we update conv3* and FC* through SGD. At the same time, for the i-th batch with $\mu \mid i$, $\mathcal{L}'_{\text{BCE}}$ is calculated by feeding all of the samples from the μ latest batches to $\tilde{D}$ and then we update conv2* through DPSGD(ε_2). An example of asymmetric training is shown in Figure 7. Here, we make an important observation that updating conv2* through DPSGD(ε_2) for every μ iterations virtually increases batch size μ times for conv2*.

Given that μ controls the batch size for updating conv2*, a natural question that arises is whether μ can be increased arbitrarily. In fact, we cannot arbitrarily increase μ because the increased μ also leads to a less frequent update of conv2*, which may, in turn, degrade the utility.

C. Discussion

Here, we discuss the rationale behind the design of DPAF.

Why Not Eliminate conv2*. Consider the case where all of the layers before the DP feature aggregation belong to conv1*. The number of learnable parameters in D will be much smaller (i.e., only conv3* and FC*); i.e., no conv2* exists. Such a setting hurts the training of GANs. This can be attributed to the fact that one knows from the GAN literature that if G (D) is much stronger than D (G), the training of GANs will likely fail to converge. In addition, conv3* and FC* might have fewer parameters compared to conv2*, depending on different model structures. It is difficult to well train D under this circumstance. Thus, keeping certain layers as conv2* is beneficial for adversarial learning.

Why Not More Layers for conv2*. As more learnable parameters in D may help the training of GANs, why conv2* does not have more layers? This can be explained as follows. If conv2* has more layers (parameters), because conv2* is updated through DPSGD, gradient clipping will lead to more information loss, flattening the feature values. In addition, if conv2* has more layers (parameters), because conv1 and conv2* both are trained by DPSGD, the output of (conv1, conv2*) will be too noisy, hindering the utility.

Why Not More Layers for conv3*. A question that may arise is why conv3* does not have more layers. As the total number of layers is fixed given an input image, if conv3* has more layers, then either conv1 or conv2* (or both) will be shrunk. Thus, DPAGG is closer to low-level features. In such a case, D cannot have meaningful learning from the aggregation of level-level features.

Choice of cGAN. The DPAF is designed to support conditional generation. Thus, one needs to consider a cGAN in DPAF. Compared to GANs, G and D of cGANs need to consider the class label to ensure both the indistinguishability between the real and synthetic samples and the consistency between the input label and the label of synthetic samples. In general, there are two straightforward solutions for label injection. First, the class label is added as part of the input vector in such a case. If we feed labels to the input layer, the labels will be diluted in the forward phase and have a weak signal only. Second, D is designed with two loss functions; one for the ordinary GAN loss and another for the class label. A representative of such a design choice is AC-GAN [51], which outputs labels as part of the loss function. Nevertheless, due to access to the label, such a design leads to a privacy budget splitting and therefore suffers from utility degradation.

In our design, DPAF follows the architecture of cDCGAN [52]. Inspired by [61] stating that the class label is better added to the first layer, we decide to use cDCGAN though there are no considerations of aggregation and DP in [61]. In essence, cDCGAN feeds labels to the second layer by first computing the embedding (from scalar to vector) of labels, significantly strengthening the signal. A natural question that arises is why the class label is not added to the latter layers of D, given the class label in the latter layers may preserve an even stronger signal. The drawback of doing so is that all the layers before the layer to which the class label is added can hardly learn anything, because the classifier can know the portion by looking at the label only.

The Position of DPAGG. DPAF heavily relies on DPAGG to raise the utility of synthetic samples. Thus, a natural question that arises is where the best position for DPAGG is. Without the loss of generality, DPAGG is placed to minimize the ℓ_2-sensitivity $\Delta_{2,\text{AGG}}$ in Eq. (4). As $\Delta_{2,\text{AGG}} = \sqrt{m}p$, $\Delta_{2,\text{AGG}}$ is dependent on the position of DPAGG. Given the design of a conventional CNN, where the height/width of feature maps in the next convolutional layer is half of the ones in the current convolutional layer, if the input is a $\rho \times \rho$ c-channel image, $\Delta_{2,\text{AGG}}$ can be calculated as $\sqrt{c \prod_{i=1}^{a} k_i} \cdot \frac{\rho}{2^a}$, where k_i is the number of filters in the i-th convolutional layer and DPAGG is placed behind the a-th convolutional layer. Define R_j as $\left(\sqrt{c \prod_{i=1}^{j} k_i} \cdot \frac{\rho}{2^j}\right) / \left(\sqrt{c \prod_{i=1}^{j-1} k_i} \cdot \frac{\rho}{2^{j-1}}\right)$. We can easily derive $R_j = \sqrt{k_j}/2$. Thus, if $k_j \geq 4$, then $\Delta_{2,\text{AGG}}$ is monotone increasing from earlier to latter layers. As a consequence, from the ℓ_2-sensitivity point of view, we conclude that the best position for DPAGG is between the first and second convolutional layers. Unfortunately, placing DPAGG in such a position does not lead to a decent utility in practice, because it completely destroys the structure of DPAF (e.g., the disappearance of conv1 and conv2*). Hence, we will empirically examine the other configurations in Section VI.

Privacy Analysis. We first prove that conv1 satisfies DP in Corollary 1 in Section VII-C of the Supplementary Materials. Afterward, we prove that DPAF satisfies DP in Theorem 6 in Section VII-C of the Supplementary Materials. The formal description of all the theorems, together with their mathmatical proofs, can be found in the Supplementary Materials.

VI. Experiment Evaluation

In this section, we present the experiment evaluation of DPAF. We chose representative datasets for the task of data synthesis. After that, DPAF was conducted to generate synthetic data. We then evaluated the utility of the synthetic data based on different settings of hyperparameters.

A. Experiment Setup

We describe the datasets, baselines, evaluation metrics, and architecture of our canonical implementation of DPAF below.

Dataset. In our experiments, we focus on high-reslution image synthesis. We considered CelebA and FFHQ datasets. CelebA contains colorful celebrity images of different sizes. In our experiments, we rescaled all of CelebA images into 64×64 colorful images. Based on CelebA, we created two more datasets, CelebA-Gender and CelebA-Hair, where the former is for binary classification with gender as the label and the latter is for multiclass classification dataset with hair color (black/blonde/brown) as the label. FFHQ contains 70000 128×128 colorful facial images with gender as label[2] and we created FFHQ-Gender dataset for binary classification.

We followed the default training (182637 samples) and testing sets (19962 samples) for CelebA. FFHQ contains 69471 images with 38388 females and 31083 males. We split the data into 90% and 10% for training and testing, respectively.

Baselines. We considered the baseline methods, GS-WGAN [7], DP-MERF [29], P3GM [35], DataLens [9], G-PATE [28], DP-Sinkhorn[33], DP-HP [32], Nonlinear DPDC (NDPDC) [39], and PEARL [34]. The implementation of all the baselines is based on the official code (see Section VII-A in Supplementary Materials). Though the official code is not available online, we communicated with the authors of PEARL to have a copy.

Most of the official codes are written for synthesizing images of low resolutions. As we mainly focus on the synthesis of high-dimensional images, we made necessary modifications such as batch size and input size to make them adaptable to different settings.

Evaluation Metrics. Given two levels of privacy guarantee, $(1, 10^{-5})$-DP and $(10, 10^{-5})$-DP, we aim to evaluate the utility of DP image synthesis. The utility can have two dimensions; i.e., the classification accuracy and the visual quality. In the former case, we calculate the predicting accuracy of the classifier trained by synthetic images and tested by real images. The architecture of the classifier used in our experiment is the same as the one used in GS-WGAN, G-PATE, and DataLens and is shown in Figure 8. We conducted necessary modifications on the code for DP-MERF, P3GM, DP-Sinkhorn, DP-HP, NDPDC, and PEARL to derive their accuracies under the same setting. This explains the inconsistency between the results in this paper and the results reported in the original papers. On the other hand, in the latter case, we display the synthetic images for visualization and report Fréchet inception distance (FID).

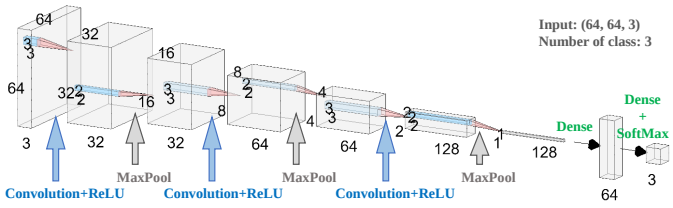


Fig. 8: The architecture of the evaluated classifier

Canonical Implementation of DPAF. Basically, DPAF adds the DP feature aggregation on the basis of cDCGAN. In our canonical implementation, the batch size is 64 for CelebA and FFHQ. The latent vector sampled from the standard Gaussian distribution is of dimension 100. The asymmetry multiplier $\mu = 8$. We also apply gradient compression [62] to the per-sample gradient to keep the top 90% values only. Our DPAF is configured to be C2-C2-C1 for CelebA and C3-C1-× for FFHQ, where the notation Cx_1-Cx_2-Cx_3 means that the D uses x_1 layers as conv1, x_2 layers as conv2*, and x_3 layers as conv3*. The notation × means that the corresponding layer does not exist. We always have two FC* layers.

For the privacy budget allocation, the notation $(x_1\%, x_2\%, x_3\%)$ refers to the setting, where conv1, conv2*, and DPAGG have $\varepsilon_1 = \frac{x_1 \cdot \varepsilon}{100}$, $\varepsilon_2 = \frac{x_2 \cdot \varepsilon}{100}$, and $\varepsilon_3 = \frac{x_3 \cdot \varepsilon}{100}$, respectively, given the total privacy budget ε. A similar notation is $(x_1, \times, x_3)$, where both conv1 and DPAGG have a privacy budget $\varepsilon_1 = x_1$ and $\varepsilon_3 = x_3$, respectively, and conv2* has the rest. For example, $(0.1, \times, 0.1)$ means that conv1, conv2, and DPAGG have 0.1, 9.8, and 0.1, respectively, if the total privacy budget is 10. Throughout Section VI-B, *canonical accuracy* means the accuracy from the canonical implementation.

B. Experiment Results

Here, we present our experiment results and ablation study. All of the experiment results below are derived by averaging the results from five independent experiments.

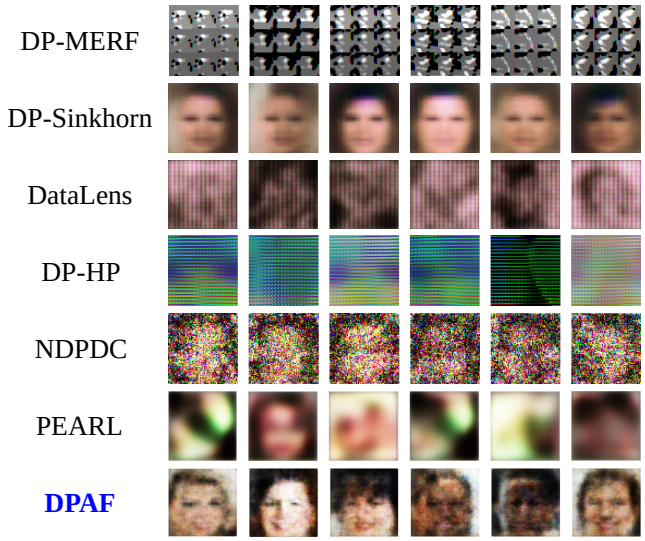


Fig. 9: Visual results for 64×64 CelebA-Gender with $\varepsilon = 10$. The left (right) three columns are females (males).

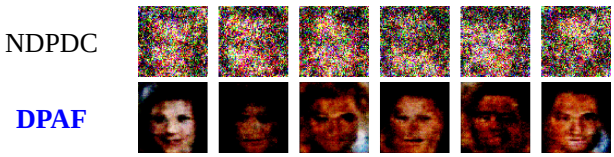


Fig. 10: Visual results for 64×64 CelebA-Gender images with $\varepsilon = 1$. The left (right) three columns are females (males).

[2]FFHQ labels from https://github.com/DCGM/ffhq-features-dataset/tree/master/json.

	ε	DataLens	DP-HP	NDPDC	PEARL	DPAF
CelebA	$\varepsilon = 1$	298	352	**235**	303	285
-Gender	$\varepsilon = 10$	320	341	**183**	302	298
CelebA	$\varepsilon = 1$	×	339	**239**	338	301
-Hair	$\varepsilon = 10$	×	340	**185**	337	298

TABLE I: The comparison of FIDs for 64×64 CelebA.

1) Visual Quality: We first present the results of the visual quality evaluation in Figure 9. In particular, the DPAF-synthesized images appear more realistic and capture more facial features, such as eyes, lips, and facial shape, compared to the uniform faces generated by DP-Sinkhorn and the highly noisy faces generated by the other baselines. Such a gain is due to the use of SIN and our design of DPAGG, i.e., the aggregated feature is more robust to the DP noise and better able to discriminate features after training.

We also present the quantitative results in Table I. We can see that NDPDC achieves the lowest FIDs, while DPAF achieves the second lowest FIDs. However, such results are inconsistent with the visual quality results in Figure 9 ($\varepsilon = 10$), because obviously DPAF-synthesized images are more realistic than NDPDC-synthesized ones. Figure 10 ($\varepsilon = 1$) also shows the same phenomenon. While FID has been widely used to evaluate the quality of synthesized images, this shows a limitation of FID and can be attributed to the inadequacy of FID as a perceptual metric [63], [64].

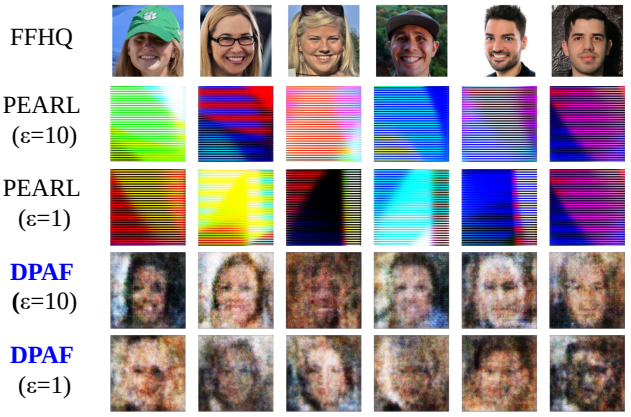


Fig. 11: Visual results for 128×128 FFHQ images. The left (right) three columns are females (males).

2) Classification Accuracy: Table II shows the classification results of DPAF and the other baseline methods. One can see from Table II that DPAF outperforms all other baselines for CelebA-Gender and CelebA-Hair. We observe that the larger image size implies more convolutional layers in DPAF, which strengthens the generative capability. In this sense, our design may suggest the potential of DPAF to synthesize images with higher resolutions, since a generator with more layers can be employed.

To further demonstrate the advantage of DPAF in synthesizing high-utility and high-dimensional images, we conducted experiments on FFHQ-Gender, and the visual results are shown in Figure 11, where the images synthesized by PEARL look like pure noise, while the images synthesized by DPAF still preserve facial features. Table III reports the predicting accuracy on FFHQ-Gender. In the above, we claim that the generative ability of DPAF increases with image size. At first glance, we can see from Table III that the accuracy gets worse compared to CelebA-Gender in Table II. However, this can be explained as follows. First, CelebA-Gender and FFHQ-Gender are two different datasets with different distributions. The direct comparison between the accuracy of CelebA-Gender and FFHQ-Gender remains doubtful. Second, synthesizing 128×128 images has reached the limit of conventional GANs without using modern techniques such as residual blocks [65]. Synthesizing higher resolution images requires much more sophisticated GANs (e.g., PGGAN [66]). Applying the techniques in DPAF to modern GANs remains unexplored, but would be our future research direction.

3) Privacy Budget Allocation w/o Transfer Learning: Since we allocate a very limited privacy budget to conv1 in our canonical implementation, a natural question is whether transfer learning is necessary. In other words, if conv1 cannot learn effectively from the data, a reasonable design choice is to abandon transfer learning and invest the privacy budget in DPAGG. Below, we examine the predicting accuracy under three strategies for allocating the privacy budget in the absence of transfer learning.

Random Parameters for conv1. We considered random weights of conv1; i.e., all weights in conv1 are sampled uniformly at random from a zero-mean Gaussian distribution with a standard deviation of 0.02 and are never updated. The results are shown in Table IV, where the notation $(\times, x_3)$ means that the privacy budget $\varepsilon_3 = x_3$ is allocated to DPAGG, while the rest of the budget is allocated to conv2*. One can see that the accuracy of $(\times, 0.5)$ and $(\times, 0.2)$ is only slightly lower than the canonical accuracy. Since conv1 is supposed to learn low-level features, even if conv1 uses random features, the learning of conv2*, conv3*, and FC* can be adapted to random conv1 and perform well. However, according to our empirical experience, we still spent a very limited privacy budget on conv1 to avoid undesirable cases where some feature maps happen to contain only zero or near-zero values, rendering such feature maps useless. Higher variances of $(\times, 0.5)$ and $(\times, 0.2)$ also justify the above design choice.

Updating conv1 During Training of D. Here we did not perform transfer learning, but still used DPSGD(ε_1) to update conv1 during training of D. Table V shows the results, where the canonical implementation outperforms the other configurations. There are two reasons for this. First, while the update of conv1 is done during the training of C, since conv2, conv3, and FC in C are updated by SGD, conv1 is more informative. While conv1 is updated during training of D, conv1 is less informative due to noise accumulation. Second, in the absence of transfer learning, more layers (parameters) need to be updated during the training of D, which is more difficult to train well from an adversarial learning perspective. Note that $(0.1, \times, 0.1)$ in the second column of Table V means that we updated conv1 by DPSGD(ε_1) with $\varepsilon_1 = 0.1$ in the training of D, although the same $(0.1, \times, 0.1)$ has a different interpretation in the context of using transfer learning, as shown in Section VI-A.

Joint Updating conv1 and conv2* During Training of D. In this case, conv1 and conv2* are considered together and are updated together by the DPSGD. We saw (conv1, conv2*) as a larger component. Compared to the individual conv1 and conv2*, gradient clipping in DPSGD may cause more information loss, and DPSGD will cause more damage to the gradient structure. The results in Table VI support the above arguments.

	ε	GS-WGAN	DP-MERF	P3GM	G-PATE	DP-Sinkhorn	DataLens	DP-HP	NDPDC	PEARL	DPAF
CelebA-Gender	$\varepsilon = 1$	0.590	0.594	0.567	0.670	0.543	0.700	0.656	0.540	0.634	**0.802**
	$\varepsilon = 10$	0.614	0.608	0.588	0.690	0.621	0.729	0.617	0.600	0.646	**0.826**
CelebA-Hair	$\varepsilon = 1$	0.420	0.441	0.453	0.499	×	0.606	0.561	0.498	0.606	**0.675**
	$\varepsilon = 10$	0.523	0.449	0.486	0.622	×	0.622	0.474	0.462	0.626	**0.671**

TABLE II: Classification accuracy results under $(1, 10^{-5})$-DP and $(10, 10^{-5})$-DP. We have two ×'s because we failed to modify the code of DP-Sinkhorn and synthesize CelebA-Hair images. The rightmost column shows canonical accuracy.

	ε	PEARL	DPAF
FFHQ-Gender	$\varepsilon = 1$	0.441 ± 0.019	**0.567** ± 0.038
	$\varepsilon = 10$	0.511 ± 0.027	**0.646** ± 0.004

TABLE III: Accuracy comparison for 128×128 FFHQ-Gender.

	ε	DPAF	(×, 0.5)	(×, 0.2)
CelebA -Gender	$\varepsilon = 1$	**0.802** ± 0.018	0.752 ± 0.045	0.774 ± 0.012
	$\varepsilon = 10$	**0.826** ± 0.010	0.793 ± 0.024	0.700 ± 0.093
CelebA -Hair	$\varepsilon = 1$	**0.675** ± 0.013	0.635 ± 0.035	0.667 ± 0.029
	$\varepsilon = 10$	0.671 ± 0.014	**0.681** ± 0.015	0.670 ± 0.016

TABLE IV: The accuracy of random parameters for conv1.

	ε	DPAF	(0.1, × , 0.1)	(33%, 34%, 33%)
CelebA -Gender	$\varepsilon = 1$	**0.802** ± 0.018	0.594 ± 0.112	0.727 ± 0.143
	$\varepsilon = 10$	**0.826** ± 0.010	0.747 ± 0.100	0.817 ± 0.024
CelebA -Hair	$\varepsilon = 1$	**0.675** ± 0.013	0.424 ± 0.079	0.642 ± 0.038
	$\varepsilon = 10$	0.671 ± 0.014	0.599 ± 0.116	**0.685** ± 0.018

TABLE V: Accuracy of updating conv1 during training of D.

	ε	DPAF	(50%, 50%)	(×, 0.1)
CelebA -Gender	$\varepsilon = 1$	**0.802** ± 0.018	0.768 ± 0.036	0.748 ± 0.072
	$\varepsilon = 10$	**0.826** ± 0.010	0.759 ± 0.030	0.790 ± 0.020
CelebA -Hair	$\varepsilon = 1$	**0.675** ± 0.013	0.661 ± 0.034	0.643 ± 0.028
	$\varepsilon = 10$	**0.671** ± 0.014	0.648 ± 0.038	0.660 ± 0.018

TABLE VI: The classification accuracy of joint updating conv1 and conv2* during training of D.

Furthermore, by comparing the $(0.1, \times, 0.1)$ column in Table V and the $(\times, 0.1)$ column in Table VI (due to their similar settings), we can see that the accuracy of the former is consistently lower than that of the latter. Unlike Table VI, conv1 and con2* are treated separately in Table V, and thus suffer from privacy budget splitting, resulting in worse accuracy.

4) Privacy Budget Allocation w/ Transfer Learning: Given the regular use of DPAF (i.e., DPAF with transfer learning), we aim to examine the impact of different budget allocations on accuracy.

A general guideline for allocating privacy budgets is that the earlier (latter) layers should earn more (fewer) budgets. The rationale is that the earlier layers learn the low-level features and the latter layers will be adapted to the low-level features. Once the earlier layers have only a limited budget and the parameters fluctuate, the latter layers can hardly be adapted to the fast change of earlier layers and can hardly learn informative parameters. However, during the training of D, conv1 is frozen and does not need to be updated. In addition, as mentioned in Section VI-B3, ε_1 can be a small value. The gradient vector may have many small or even nearly zero values, which can easily be affected by the DP noise. On the other hand, the aggregated vector output by DPAGG is designed to have larger values for better robustness against the DP noise. Thus, a reasonable choice is to have $\varepsilon_2 > \varepsilon_3$. The above arguments can be confirmed empirically because we can see from Table VII that the canonical setting $(0.1, \times, 0.1)$ that follows the above discussion outperforms the other settings.

5) Number of Layers for conv1, conv2, and conv3*:* Using CelebA-Gender and CelebA-Hair as examples, we aim to know which layer configuration will result in better accuracy. As both CelebA-Gender and CelebA-Hair are 64×64, we know that there are at most five layers in total. Note that, in contrast to ordinary GANs, deliberately setting more layers in DPGANs may, in turn, hurt the training result [67] because the lengthier gradient will lead to greater information loss, failing the convergence, according to our experience. There are too many configurations to exhaustively examine. Table X shows the only results of accuracy in the cases where conv1 and conv2* jointly occupy at most four layers. From Table X, we know that C2-C1-×, C2-C2-×, C3-C1-×, and C1-C3-× result in better accuracy. Thus, given the above results, we include the consideration of conv3* in Table VIII, because Table X does not consider conv3*. The results in Table VIII support our design choice for the canonical implementation of C2-C2-C1 because it outperforms the other settings.

6) The Impact of Asymmetry Multiplier μ.: A larger μ implies a much larger budget for updating conv2*, given ε_3 for DPAGG. Obviously, the increased μ raises accuracy because conv2* virtually has more budget. Moreover, Table IX supports our claim in Section V-C that μ cannot be arbitrarily increased. The reason is that the increased μ also leads to a less frequent update of conv2*, which may, in turn, degrade the utility.

7) The Other Techniques in Enhancing Accuracy: Many techniques have been proposed to reduce the negative impact of DPSGD on model training. We examine three of them to see whether they provide similar benefits to DPAF.

Pre-Training the Model with Public Data. The recent development of DP classifiers and DPGANs has witnessed that extra data may help improve the performance of DP models [13], [21], [14]. Here, we want to examine whether pre-training the model with public data helps DPAF raise its utility. Here, the common setting in Table XI is that we follow DPAF to train C, perform transfer learning, and then train G and D on the CIFAR-10 dataset without considering DP. After that, DPAF is used to train DPGAN with the pre-trained D as D and the randomly initialized parameters as G. We additionally train DPGAN completely based on the pre-trained parameters for both G and D. One can see from Table XI that the extra data still helps the utility of DPAF.

The Impact of Gradient Compression. Gradient compression (GC) [62] is originally proposed to reduce the communication cost in federated learning. The rationale behind gradient compression is that most of the values in the gradient contribute nearly no information on the update. Different from the original case, where GC works on the gradient averaged over the samples in a batch, the canonical implementation of DPAF adopts GC to keep only the top 90% values of per-sample gradients and then performs the averaging. However,

	ε	DPAF $(0.1, \times, 0.1)$	(20%, 20%, 60%)	(20%, 60%, 20%)	(20%, 40%, 40%)	(30%, 20%, 50%)	(30%, 50%, 20%)
CelebA-Gender	$\varepsilon = 1$	**0.802** ± 0.018	0.741 ± 0.074	0.801 ± 0.035	0.793 ± 0.030	0.771 ± 0.035	0.795 ± 0.033
	$\varepsilon = 10$	**0.826** ± 0.010	0.787 ± 0.018	0.820 ± 0.020	0.790 ± 0.017	0.767 ± 0.040	0.813 ± 0.013
CelebA-Hair	$\varepsilon = 1$	0.675 ± 0.013	0.570 ± 0.144	0.663 ± 0.031	0.657 ± 0.023	0.643 ± 0.046	0.666 ± 0.024
	$\varepsilon = 10$	**0.671** ± 0.014	0.639 ± 0.031	0.619 ± 0.038	0.659 ± 0.016	0.598 ± 0.050	0.577 ± 0.094

	ε	(30%, 40%, 30%)	(30%, 30%, 40%)	(40%, 30%, 30%)	(40%, 20%, 40%)	(40%, 40%, 20%)
CelebA-Gender	$\varepsilon = 1$	0.791 ± 0.012	0.704 ± 0.164	0.745 ± 0.102	0.759 ± 0.025	0.763 ± 0.038
	$\varepsilon = 10$	0.799 ± 0.022	0.799 ± 0.023	0.797 ± 0.015	0.782 ± 0.032	0.798 ± 0.022
CelebA-Hair	$\varepsilon = 1$	**0.677** ± 0.011	0.653 ± 0.024	0.646 ± 0.039	0.454 ± 0.150	0.641 ± 0.064
	$\varepsilon = 10$	0.610 ± 0.018	0.643 ± 0.012	0.628 ± 0.011	0.634 ± 0.036	0.645 ± 0.021

TABLE VII: The classification accuracy of different privacy budget allocations with transfer learning.

	ε	DPAF (C2-C2-C1)	C1-C3-C1	C2-C1-C1	C2-C1-C2	C3-C1-C1
CelebA-Gender	$\varepsilon = 1$	**0.802** ± 0.018	0.448 ± 0.164	0.673 ± 0.099	0.505 ± 0.067	0.800 ± 0.017
	$\varepsilon = 10$	**0.826** ± 0.010	0.727 ± 0.039	0.803 ± 0.022	0.514 ± 0.091	0.820 ± 0.015
CelebA-Hair	$\varepsilon = 1$	**0.675** ± 0.013	0.356 ± 0.077	0.540 ± 0.036	0.354 ± 0.038	0.669 ± 0.018
	$\varepsilon = 10$	**0.671** ± 0.014	0.368 ± 0.074	0.664 ± 0.035	0.352 ± 0.055	0.670 ± 0.016

TABLE VIII: The classification accuracy of different layer architecture for conv1, conv2*, and conv3*.

	ε	$\mu = 2$	$\mu = 4$	DPAF ($\mu = 8$)	$\mu = 10$	$\mu = 20$
CelebA-Gender	$\varepsilon = 1$	0.775 ± 0.037	0.688 ± 0.070	**0.802** ± 0.018	0.772 ± 0.036	0.773 ± 0.030
	$\varepsilon = 10$	0.665 ± 0.162	0.819 ± 0.033	**0.826** ± 0.010	0.745 ± 0.010	0.735 ± 0.110
CelebA-Hair	$\varepsilon = 1$	0.578 ± 0.094	0.642 ± 0.038	**0.675** ± 0.013	0.656 ± 0.017	0.648 ± 0.011
	$\varepsilon = 10$	0.670 ± 0.027	0.666 ± 0.041	**0.671** ± 0.014	0.670 ± 0.015	0.668 ± 0.020

TABLE IX: The classification accuracy of different asymmetry multipliers μ's.

	ε	C1-C1-×	C1-C2-×	C2-C1-×	C1-C3-×	C2-C2-×	C3-C1-×
CelebA-Gender	$\varepsilon = 1$	0.629 ± 0.040	0.737 ± 0.025	**0.824** ± 0.025	0.661 ± 0.144	0.805 ± 0.021	0.811 ± 0.020
	$\varepsilon = 10$	0.720 ± 0.045	0.733 ± 0.038	0.762 ± 0.079	0.729 ± 0.032	**0.786** ± 0.018	0.751 ± 0.036
CelebA-Hair	$\varepsilon = 1$	0.423 ± 0.089	0.475 ± 0.120	0.643 ± 0.014	**0.662** ± 0.019	0.519 ± 0.095	0.623 ± 0.038
	$\varepsilon = 10$	0.565 ± 0.019	0.644 ± 0.024	0.639 ± 0.039	0.657 ± 0.023	**0.683** ± 0.022	0.659 ± 0.019

TABLE X: The classification accuracy of different layer architecture for conv1 and conv2*.

	ε	DPAF	Trans(D)	Trans($G + D$)
CelebA -Gender	$\varepsilon = 1$	0.802 ± 0.018	0.819 ± 0.020	**0.820** ± 0.027
	$\varepsilon = 10$	0.826 ± 0.010	0.830 ± 0.018	**0.831** ± 0.023
CelebA -Hair	$\varepsilon = 1$	0.675 ± 0.013	0.663 ± 0.038	**0.695** ± 0.015
	$\varepsilon = 10$	0.671 ± 0.014	0.695 ± 0.036	**0.703** ± 0.015

TABLE XI: The accuracy of DPAF with public data pre-training.

	ε	DPAF	w/o GC	TOPAGG [9]
CelebA -Gender	$\varepsilon = 1$	**0.802** ± 0.018	0.725 ± 0.150	0.549 ± 0.075
	$\varepsilon = 10$	**0.826** ± 0.010	0.818 ± 0.022	0.530 ± 0.097
CelebA -Hair	$\varepsilon = 1$	**0.675** ± 0.013	0.673 ± 0.016	0.359 ± 0.079
	$\varepsilon = 10$	0.671 ± 0.014	**0.678** ± 0.017	0.375 ± 0.056

TABLE XII: Acc of DPAF with different compression strategies.

	ε	DPAF	w/ TS
CelebA-Gender	$\varepsilon = 1$	**0.802** ± 0.018	**0.802** ± 0.015
	$\varepsilon = 10$	**0.826** ± 0.010	0.806 ± 0.025
CelebA-Hair	$\varepsilon = 1$	**0.675** ± 0.013	0.594 ± 0.031
	$\varepsilon = 10$	**0.671** ± 0.014	0.620 ± 0.043

TABLE XIII: Acc of DPAF with tempered sigmoid (TS) functions.

we still want to examine whether GC can help DPSGD. The comparison between the DPAF column and "w/o GC" column in Table XII still shows that DPSGD can benefit from GC because the information loss from gradient clipping can be mitigated. TOPAGG [9] is a modified DPSGD that works on compressed and quantized gradients. The GC in TOPAGG is configurated to keep the top-k values only[3]. Nevertheless, TOPAGG gains lower accuracy. This can be explained by considering the design of TOPAGG. In particular, the success of TOPAGG, in essence, relies on training a large number of teacher classifiers[9], [33]. As DPAF does not fit such a requirement, TOPAGG on DPAF does not perform well.

Tempered Sigmoid Activation Function. Papernot et al. [68] find that exploding activations cause the unclipped gradient magnitude to increase and therefore gradient clipping leads to more information loss. Thus, tempered sigmoid (TS) [68], a family of activation functions, is proposed to replace the conventional activation functions in DPSGD. Table XIII shows the results, where hyperbolic tangent (tanh), as a representative of TS, is used to replace the leaky ReLU in our canonical DPAF. In our test, tanh is used in DPAF, and we can see from Table XIII that it, in turn, leads to worse accuracy. This can be explained as follows. First, Papernot et al. [68] conduct the experiments on DP classifiers only. Whether TS can raise the utility of DPGANs remains unknown. Second, a bounded activation (e.g., sigmoid and tanh) easily causes gradient vanishing and therefore is rarely used in practice. On the contrary, the unbounded ones (e.g., ReLU and leaky ReLU) are more capable of avoiding gradient vanishing [52]. conv1 and conv2* in DPAF are updated by DPSGD, but conv3* and FC* are updated by SGD. While TS is beneficial to DPSGD (for conv1 and conv2*) but harmful to SGD (conv2* and FC*). Overall, adopting tanh in DPAF slightly degrades utility.

[3]For CelebA-Gender, $k = 200$ with $\varepsilon = 1$ and $k = 3000$ with $\varepsilon = 10$. For CelebA-Hair, $k = 150$ with $\varepsilon = 1$ and $k = 200$ with $\varepsilon = 10$.

8) Comparison to Private-GANs: A concurrent work, Private-GANs [69], is conceptually similar to but can be seen as an oversimplified version of DPAF. Tables XIV, XV, and XVI show the comparison between DPAF and Private-GANs. In particular, the results of DPAF in Table XIV are derived by optimizing both μ and n_{critic} (called n_D in their paper). As Private-GANs reported the results with $n_{\text{critic}} = 1, 10, 50, 100, 200$, to be aligned with their results, Table XV and XVI show the results of DPAF and Private-GANs for $n_{\text{critic}} = 1, 10, 50, 100, 200$. One can see from Tables XIV, XV, and XVI that DPAF outperforms Private-GANs. This can be attributed to the fact that introducing μ in the design of DPAF as asymmetric training significantly reduces the downside from raising n_{critic}.

	ε	NDPDC	PEARL	Private-GANs	DPAF
CelebA-Gender	0.5	0.518	0.655	0.620	**0.768**
	1	0.540	0.634	0.663	**0.802**
	5	0.535	0.639	0.679	**0.789**
	10	0.600	0.646	0.714	**0.826**
CelebA-Hair	0.5	0.497	0.592	0.513	**0.663**
	1	0.498	0.606	0.474	**0.675**
	5	0.469	0.609	0.508	**0.680**
	10	0.462	0.626	0.540	**0.671**

TABLE XIV: More classification results. Each value is derived by averaging the results from 50 independent trials.

	ε	$n_{\text{critic}} = 1$	$n_{\text{critic}} = 10$	$n_{\text{critic}} = 50$	$n_{\text{critic}} = 100$	$n_{\text{critic}} = 200$
Private -GANs	0.5	0.468	0.448	0.612	0.620	0.598
	1	0.570	0.541	0.629	0.663	0.498
	5	0.584	0.572	0.652	0.680	0.634
	10	0.607	0.673	0.677	0.714	0.623
DPAF ($\mu = 1$)	0.5	0.490	0.650	0.511	0.463	0.515
	1	0.507	0.666	0.491	0.466	0.549
	5	0.535	0.690	0.500	0.524	0.496
	10	0.503	0.666	0.552	0.452	0.546
DPAF ($\mu = 8$)	0.5	0.501	**0.668**	0.579	0.500	0.517
	1	0.567	**0.725**	0.573	0.537	0.510
	5	0.687	**0.715**	0.524	0.511	0.503
	10	**0.789**	0.690	0.556	0.519	0.472

TABLE XV: The classification accuracy of different n_{critic}'s on CelebA-Gender. Bold numbers (e.g., 0.668) are the largest among all values with the same ε (e.g., $\varepsilon = 0.5$).

	ε	$n_{\text{critic}} = 1$	$n_{\text{critic}} = 10$	$n_{\text{critic}} = 50$	$n_{\text{critic}} = 100$	$n_{\text{critic}} = 200$
Private -GANs	0.5	0.513	0.441	0.488	0.411	0.356
	1	0.474	0.437	0.366	0.355	0.357
	5	0.508	0.498	0.416	0.408	0.405
	10	0.540	0.491	0.466	0.454	0.373
DPAF ($\mu = 1$)	0.5	0.367	0.533	0.368	0.356	0.331
	1	0.301	0.467	0.358	0.372	0.333
	5	0.353	0.518	0.357	0.332	0.345
	10	0.379	0.523	0.351	0.375	0.324
DPAF ($\mu = 8$)	0.5	**0.616**	0.540	0.366	0.343	0.306
	1	0.345	**0.514**	0.361	0.345	0.319
	5	0.468	**0.524**	0.329	0.367	0.319
	10	**0.556**	0.513	0.357	0.347	0.308

TABLE XVI: The classification accuracy of different n_{critic}'s on CelebA-Hair. Bold numbers (e.g., 0.616) are the largest among all values with the same ε (e.g., $\varepsilon = 0.5$).

9) Empirical Evidence for Data Privacy of DPAF against MIA: GAN-Leaks [1] offers a tool to evaluate whether a candidate GAN-synthesized dataset can resist MIA. Here, in addition to a formal privacy proof in Section VII-C of Supplementary Materials, we provide additional empirical evidence in Table XVII that the DPAF-synthesized dataset can resist MIA. In particular, each value in Table XVII refers to the AUC-ROC (area under the curve of ROC) of MIA in identifying members and non-members. One can see from Table XVII that if the synthetic dataset is generated by a model trained by ordinary SGD, then the AUC-ROC is 1, which means that MIA is always successful. On the contrary, in the case of the synthetic dataset from DPAF, the AUC-ROC is nearly 0.5, failing MIA.

		64 Images	128 Images
SGD		1.0	1.0
DPAF	$\epsilon = 10$	0.466	0.502
	$\epsilon = 5$	0.469	0.499
	$\epsilon = 1$	0.473	0.499
	$\epsilon = 0.5$	0.479	0.500

TABLE XVII: AUC-ROC of MIA on CelebA-Gender with DPAF. The sizes of the training set is 64 and 128 (aligned with the setting used in [1]).

VII. Conclusion

Overall, we propose a novel and effective DPGAN, DPAF, which can synthesize high-dimensional image data. Fundamentally different from the prior works, DPAF is featured by the DP feature aggregation in the forward phase, which significantly improves the robustness against noise. In addition, we propose a novel asymmetric training strategy, which determines an ideal batch size. We formally prove the privacy of DPAF. Extensive experiments demonstrate superior performance compared to the previous state-of-the-art methods.

References

[1] D. Chen, N. Yu, Y. Zhang, and M. Fritz, "Gan-leaks: A taxonomy of membership inference attacks against generative models," in *ACM Conference on Computer and Communications Security (CCS)*, 2020.

[2] J. Hayes, L. Melis, G. Danezis, and E. De Cristofaro, "Logan: Membership inference attacks against generative models," in *Proceedings on Privacy Enhancing Technologies (PoPETs)*, 2019.

[3] B. Hilprecht, M. Härterich, and D. Bernau, "Monte carlo and reconstruction membership inference attacks against generative models," in *Proceedings on Privacy Enhancing Technologies (PoPETs)*, 2019.

[4] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy." *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407, 2014.

[5] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," *ACM Conference on Computer and Communications Security (CCS)*, 2016.

[6] E. Bagdasaryan, O. Poursaeed, and V. Shmatikov, "Differential privacy has disparate impact on model accuracy," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[7] D. Chen, T. Orekondy, and M. Fritz, "Gs-wgan: A gradient-sanitized approach for learning differentially private generators," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[8] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International Conference on Machine Learning (ICML)*, 2017.

[9] B. Wang, F. Wu, Y. Long, L. Rimanic, C. Zhang, and B. Li, "Datalens: Scalable privacy preserving training via gradient compression and aggregation," *ACM Conference on Comp'uter and Communications Security (CCS)*, 2021.

[10] Z. Zhang, T. Wang, J. Honorio, N. Li, M. Backes, S. He, J. Chen, and Y. Zhang, "Privsyn: Differentially private data synthesis," in *USENIX Security Symposium*, 2021.

[11] R. Mckenna, D. Sheldon, and G. Miklau, "Graphical-model based estimation and inference for differential privacy," in *International Conference on Machine Learning (ICML)*, 2019.

[12] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao, "Privbayes: Private data release via bayesian networks," *ACM Transactions on Database Systems*, vol. 42, no. 4, oct 2017.

[13] S. De, L. Berrada, J. Hayes, S. L. Smith, and B. Balle, "Unlocking high-accuracy differentially private image classification through scale," arXiv: 2204.13650, 2022. [Online]. Available: https://arxiv.org/pdf/2204.13650.pdf

[14] F. Tramèr and D. Boneh, "Differentially private learning needs better features (or much more data)," in *International Conference on Learning Representations (ICLR)*, 2021.

[15] D. Yu, H. Zhang, W. Chen, J. Yin, and T.-Y. Liu, "Large scale private learning via low-rank reparametrization," in *International Conference on Machine Learning (ICML)*, 2021.

[16] D. Yu, S. Naik, A. Backurs, S. Gopi, H. A. Inan, G. Kamath, J. Kulkarni, Y. T. Lee, A. Manoel, L. Wutschitz, S. Yekhanin, and H. Zhang, "Differentially private fine-tuning of language models," in *International Conference on Learning Representations (ICLR)*, 2022.

[17] X. Li, F. Tramèr, P. Liang, and T. Hashimoto, "Large language models can be strong differentially private learners," in *International Conference on Learning Representations (ICLR)*, 2022.

[18] N. Papernot, M. Abadi, Úlfar Erlingsson, I. Goodfellow, and K. Talwar, "Semi-supervised knowledge transfer for deep learning from private training data," in *International Conference on Learning Representations (ICLR)*, 2017.

[19] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and Ú. Erlingsson, "Scalable private learning with pate," in *International Conference on Learning Representations (ICLR)*, 2018.

[20] J. Jordon, J. Yoon, and M. van der Schaar, "Pate-gan: Generating synthetic data with differential privacy guarantees," in *International Conference on Learning Representations (ICLR)*, 2019.

[21] X. Zhang, S. Ji, and T. Wang, "Differentially private releasing via deep generative model," in *arXiv: 1801.01594*, 2018. [Online]. Available: https://arxiv.org/abs/1801.01594

[22] H. B. McMahan and G. Andrew, "A general approach to adding differential privacy to iterative training procedures," *NeurIPS Workshop on Privacy Preserving Machine Learning (PPML)*, 2018.

[23] O. Thakkar, G. Andrew, and H. B. McMahan, "Differentially private learning with adaptive clipping," *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[24] C. Xu, J. Ren, D. Zhang, Y. Zhang, Z. Qin, and R. Kui, "Ganobfuscator: Mitigating information leakage under gan via differential privacy," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 9, pp. 2358–2371, 2019.

[25] L. Xie, K. Lin, S. Wang, F. Wang, and J. Zhou, "Differentially private generative adversarial network," arXiv: 1802.06739, 2018. [Online]. Available: https://arxiv.org/pdf/1802.06739.pdf

[26] D. Yu, H. Zhang, W. Chen, and T. Liu, "Do not let privacy overbill utility: Gradient embedding perturbation for private learning," *International Conference on Learning Representations (ICLR)*, 2021.

[27] M. Nasr, R. Shokri, and A. Houmansadr, "Improving deep learning with differential privacy using gradient encoding and denoising," in *Theory and Practice of Differential Privacy*, 2020.

[28] Y. Long, B. Wang, Z. Yang, B. Kailkhura, A. Zhang, C. A. Gunter, and B. Li, "G-pate: Scalable differentially private data generator via private aggregation of teacher discriminators," *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[29] F. Harder, K. Adamczewski, and M. Park, "Dp-merf: Differentially private mean embeddings with random features for practical privacy-preserving data generation," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021.

[30] F. Harder, M. Jalali, D. J. Sutherland, and M. Park, "Pre-trained perceptual features improve differentially private image generation," *Transactions on Machine Learning Research (TMLR)*, 2023.

[31] Y. Yang, K. Adamczewski, D. J. Sutherland, X. Li, and M. Park, "Differentially private neural tangent kernels for privacy-preserving data generation," in *AAAI Workshop on Privacy-Preserving Artificial Intelligence (PPAI-24)*, 2024.

[32] M. Vinaroz, M.-A. Charusaie, F. Harder, K. Adamczewski, and M. J. Park, "Hermite polynomial features for private data generation," in *International Conference on Machine Learning (ICML)*, 2022.

[33] T. Cao, A. Bie, A. Vahdat, S. Fidler, and K. Kreis, "Don't generate me: Training differentially private generative models with sinkhorn divergence," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[34] S. P. Liew, T. Takahashi, and M. Ueno, "Pearl: Data synthesis via private embeddings and adversarial reconstruction learning," in *International Conference on Learning Representations (ICLR)*, 2022.

[35] S. Takagi, T. Takahashi, Y. Cao, and M. Yoshikawa, "P3gm: Private high-dimensional data release via privacy preserving phased generative model," in *IEEE International Conference on Data Engineering (ICDE)*, 2021.

[36] D. Jiang, G. Zhang, M. Karami, X. Chen, Y. Shao, and Y. Yu, "Dp^2-vae: Differentially private pre-trained variational autoencoders," 2022. [Online]. Available: https://arxiv.org/abs/2208.03409

[37] B. Pfitzner and B. Arnrich, "Dpd-fvae: Synthetic data generation using federated variational autoencoders with differentially-private decoder," 2022. [Online]. Available: https://arxiv.org/abs/2211.11591

[38] J.-W. Chen, C.-M. Yu, C.-C. Kao, T.-W. Pang, and C.-S. Lu, "Dp-gen: Differentially private generative energy-guided network for natural image synthesis," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[39] T. Zheng and B. Li, "Differentially private dataset condensation," in *Workshop on AI Systems with Confidential Computing (AISCC)*, 2024.

[40] B. Zhao, K. R. Mopuri, and H. Bilen, "Dataset condensation with gradient matching," in *International Conference on Learning Representations (ICLR)*, 2021.

[41] T. Dockhorn, T. Cao, A. Vahdat, and K. Kreis, "Differentially private diffusion models," *Transactions on Machine Learning Research*, 2023.

[42] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[43] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[44] S. Ghalebikesabi, L. Berrada, S. Gowal, I. Ktena, R. Stanforth, J. Hayes, S. De, S. L. Smith, O. Wiles, and B. Balle, "Differentially private diffusion models generate useful synthetic images," in *International Workshop on Trustworthy Federated Learning*, 2023.

[45] H. Wang, S. Pang, Z. Lu, Y. Rao, Y. Zhou, and M. Xue, "dp-promise: Differentially private diffusion probabilistic models for image synthesis," in *USENIX Security Symposium*, 2024.

[46] K. Li, C. Gong, Z. Li, Y. Zhao, X. Hou, and T. Wang, "{PrivImage}: Differentially private synthetic image generation using diffusion models with {Semantic-Aware} pretraining," in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 4837–4854.

[47] Z. Lin, S. Gopi, J. Kulkarni, H. Nori, and S. Yekhanin, "Differentially private synthetic data via foundation model apis 1: Images," in *International Conference on Learning Representations (ICLR)*, 2024.

[48] D. Kifer and A. Machanavajjhala, "No free lunch in data privacy," in *ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2011.

[49] V. Doroshenko, B. Ghazi, P. Kamath, R. Kumar, and P. Manurangsi, "Connect the dots: Tighter discrete approximations of privacy loss distributions," in *Proceedings on Privacy Enhancing Technologies (PoPETs)*, 2022.

[50] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014. [Online]. Available: https://arxiv.org/abs/1411.1784

[51] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier GANs," in *International Conference on Machine Learning (ICML)*, 2017.

[52] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *International Conference on Learning Representations (ICLR)*, 2016.

[53] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *IEEE Symposium on Security and Privacy (S&P)*, 2017.

[54] J. Ye, A. Maddi, S. K. Murakonda, V. Bindschaedler, and R. Shokri, "Enhanced membership inference attacks against machine learning models," in *ACM Conference on Computer and Communications Security (CCS)*, 2022.

[55] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *ACM Conference on Computer and Communications Security (CCS)*, 2015.

[56] N. Carlini, S. Deng, S. Garg, S. Jha, S. Mahloujifar, M. Mahmoody, S. Song, A. Thakurta, and F. Tramer, "Is private learning possible with instance encoding?" in *IEEE Symposium on Security and Privacy (S&P)*, 2021.

[57] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[58] ——, "Instance normalization: The missing ingredient for fast stylization," 2016. [Online]. Available: https://arxiv.org/abs/1607.08022

[59] X. Jin, C. Lan, W. Zeng, Z. Chen, and L. Zhang, "Style normalization and restitution for generalizable person re-identification," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

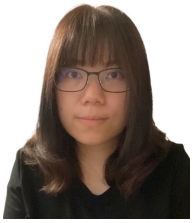
[60] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *International Conference on Computer Vision (ICCV)*, 2017.

[61] G. Perarnau, J. van de Weijer, B. Raducanu, and J. M. Álvarez, "Invertible conditional gans for image editing," in *NeurIPS Workshop on Adversarial Training*, 2016.

[62] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep Gradient Compression: Reducing the communication bandwidth for distributed training," in *The International Conference on Learning Representations (ICLR)*, 2018.

[63] M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton, "Demystifying mmd gans," in *International Conference on Learning Representations (ICLR)*, 2018.
[64] Y. Benny, T. Galanti, S. Benaim, and L. Wolf, "Evaluation metrics for conditional image generation," *International Journal of Computer Vision*, vol. 129, p. 1712–1731, 2021.
[65] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
[66] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," in *International Conference on Learning Representations (ICLR)*, 2018.
[67] R. Bassily, A. Smith, and A. Thakurta, "Private empirical risk minimization: Efficient algorithms and tight error bounds," in *IEEE 55th Annual Symposium on Foundations of Computer Science (FOCS)*, 2014.
[68] N. Papernot, A. Thakurta, S. Song, S. Chien, and Ú. Erlingsson, "Tempered sigmoid activations for deep learning with differential privacy," *AAAI Conference on Artificial Intelligence (AAAI)*, 2021.
[69] A. Bie, G. Kamath, and G. Zhang, "Private gans, revisited," *Transactions on Machine Learning Research*, 2023.
[70] I. Mironov, "Rényi differential privacy," *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pp. 263–275, 2017.
[71] Y.-X. Wang, B. Balle, and S. Kasiviswanathan, "Subsampled rényi differential privacy and analytical moments accountant," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019.

Chih-Hsun Lin is currently a Ph.D. student at Department of Computer Science, National Yang Ming Chiao Tung University. His research interests include network security and data privacy.



Chiay-Yi Hsu is currently a Ph.D. student at Department of Computer Science, National Yang Ming Chiao Tung University. She had academic visits at IBM Thomas J. Watson research center and CISPA Helmholtz Center for Information Security. Her research interests include trustworthy AI and data privacy.

Chia-Mu Yu is currently an associate professor at National Yang Ming Chiao Tung University, Taiwan. He had academic visits at IBM Thomas J. Watson research center, Harvard University, Imperial College London, University of Padova, and the University of Illinois at Chicago. He received Hwa Tse Roger Liang Junior Chair Professor, MOST Yong Scholar Fellowship, ACM/IICM K. T. Li Young Researcher Award, Observational Research Scholarship from Pan Wen Yuan Foundation, and MOST Project for Excellent Junior Research Investigators, Taiwan. He serves as an Associate Editor for IEEE Transactions on Information Forensics and Security, IEEE Internet of Things Journal, and IEEE Consumer Electronics Magazine.

Yang Cao is an Associate Professor at the Department of Computer Science, Institute of Science Tokyo (Science Tokyo, formerly Tokyo Tech), and directing the Trustworthy Data Science and AI (TDSAI) Lab. He is passionate about studying and teaching on algorithmic trustworthiness in data science and AI. Two of his papers on data privacy were selected as best paper finalists in top-tier conferences IEEE ICDE 2017 and ICME 2020. He was a recipient of the IEEE Computer Society Japan Chapter Young Author Award 2019, Database Society of Japan Kambayashi Young Researcher Award 2021. His research projects were/are supported by JSPS, JST, MSRA, KDDI, LINE, WeBank, etc.

Chun-Ying Huang joined National Yang Ming Chiao Tung University in 2016 as an Associate Professor and has been a Professor in the Department of Computer Science since 2018. His research interests span system security, multimedia networking, and mobile computing. Dr. Huang is a member of the ACM and the IEEE.

SUPPLEMENTARY MATERIALS

A. Sources of Official Code for Baseline Methods

The official code of GS-WGAN, DP-MERF, DataLens, G-PATE, DP-Sinkhorn, DP-HP, and DPDC can be found at https://github.com/DingfanChen/GS-WGAN, https://github.com/ParkLabML/DP-MERF, https://github.com/AI-secure/DataLens, https://github.com/AI-secure/G-PATE, and https://github.com/nv-tlabs/DP-Sinkhorn_code, https://github.com/ParkLabML/DP-HP, and https://openreview.net/attachment?id=H8XpqEkbua_&name=supplementary_material respectively.

B. Notation Table

The notation table summarizing the frequently used notations can be found in Table XVIII.

Symbol	Description
$\mathcal{D}$, $\mathcal{D}'$	The neighboring data
C	The classifier in DAF before transfer learning
G	The generator in DAF after transfer learning
D	The discriminator in DAF after transfer learning
μ	Asymmetry multiplier
n_{critic}	Number of critic iterations per generator iteration
ϵ	The privacy loss
δ	The probability of violating DP
σ^2	The variance of Gaussian distribution
M_f	The feature extractor (FE)
M_c	The label predictor
θ_f	The parameters of the FE
θ_c	The parameters of the label predictor
u	The clipping threshold (sensitivity of DPSGD)
clip_u	Gradient clipping function with threshold u
w	The model parameter
p	The size of feature map is $p \times p$
m	The number of feature maps
$\mathbb{B}$	The number of batches
IN	The instance normalization
SIN	The simplified instance normalization
$\mu_{i_1 i_2}$	The mean of feature map $X_{i_1 i_2}$
$\sigma^2_{i_1 i_2}$	The variance of feature map $X_{i_1 i_2}$
H	The height of the feature map
W	The width of the feature map
$x_{i_1 i_2 i_3 i_4}$	The element of feature map $X_{i_1 i_2}$
$\widehat{x_{i_1 i_2 i_3 i_4}}$	The new value of $x_{i_1 i_2 i_3 i_4}$ after SIN
α	The order in Rényi DP
D_α	The Rényi divergence of order α
G_σ	The Gaussian mechanism with variance σ^2
γ	The subsampling rate

TABLE XVIII: Notation Table

C. Privacy Analysis

In the following, we are aimed to prove that conv1 satisfies DP. We start from Theorem 1.

Theorem 1: Define a model $M(\mathcal{D}) = M_2(\theta_2, M_1(\theta_1, \mathcal{D})) : \mathcal{D} \to R_2$, where $\mathcal{D}$ is the sensitive data, θ_1 and θ_2 are model parameters, $M_1(\theta_1, \mathcal{D}) : \mathcal{D} \to R_1$ is the first half of the model M, $M_2(\theta_2, R_1) : R_1 \to R_2$ is the second half of the model M, with R_1 and R_2 denoting the corresponding outputs of the layers. $M_1(\theta_1, \mathcal{D})$ satisfies DP if it is trained by DPSGD.

Proof 1: Define $G(\theta_1, \theta_2, \mathcal{D}, \mathcal{L}(\theta_1, \theta_2, \mathcal{D}))$ as the gradient of the model M, where $\mathcal{L}$ is the loss function. $G(\theta_1, \theta_2, \mathcal{D}, \mathcal{L}(\theta_1, \theta_2, \mathcal{D}))$ can be rewritten as

$$G(\theta_1, \theta_2, \mathcal{D}, \mathcal{L}(\theta_1, \theta_2, \mathcal{D})) \tag{5}$$

$$=[g(\theta_2, \mathcal{L}(\theta_1, \theta_2, \mathcal{D})), \tilde{g}(\theta_1, g(\theta_2, \mathcal{L}(\theta_1, \theta_2, \mathcal{D}))], \tag{6}$$

where $g(\theta_2, \mathcal{L}(\theta_1, \theta_2, \mathcal{D}))$ can be seen as the gradient of M_2 and $\tilde{g}(\theta_1, \mathcal{L}(\theta_1, \theta_2, \mathcal{D}))$ can be seen as the gradient of M_1. The updating rule is shown below.

$$\theta_2 \leftarrow \theta_2 + g(\theta_2, \mathcal{L}(\theta_1, \theta_2, \mathcal{D})) \tag{7}$$

$$\theta_1 \leftarrow \theta_1 + \tilde{g}(\theta_1, g(\theta_2, \mathcal{L}(\theta_1, \theta_2, \mathcal{D}))) \tag{8}$$

To simplify the notations, we use $\mathbb{D}$ to denote the dependency of the sensitive data. In this case, Eq. (5) can be rewritten as

$$G(\theta_1, \theta_2, \mathcal{D}, \mathcal{L}(\theta_1, \theta_2, \mathcal{D}))$$

$$=G(\theta_1, \theta_2, \mathbb{D}) \tag{9}$$

$$=[g(\theta_2, \mathbb{D}), \tilde{g}(\theta_1, g(\theta_2, \mathbb{D}))]. \tag{10}$$

As $\tilde{g}(\theta_1, \mathbb{D})$ is trained by DPSGD, one can ensures that $\tilde{g}(\theta_1, \mathbb{D})$ satisfies DP. As $M_1(\theta_1, \mathcal{D})$ is initialized randomly and is updated by $\tilde{g}(\theta_1, \mathbb{D})$, one can ensures that $M_1(\theta_1, \mathcal{D})$ satisfies DP.

With Theorem 1, we can easily conclude that conv1 in DPAF satisfies DP via Corollary 1.

Corollary 1: The conv1 in DPAF satisfies DP.

Proof 2: The classifier C in DPAF before the transfer learning is an instantiation of the model M in Theorem 1. In this case, conv1 can be seen as M_1, and conv2, conv3, and FC are seen as M_2 in Theorem 1. As shown in Line 4 of Algorithm 1, we perform SGD for updating conv2, conv2, and FC, and perform DPSGD(ε_1) for updating conv1. Hence, conv1 satisfies DP.

Next, we are aimed to prove that DPAF satisfies DP. We rely on Rényi DP (RDP) [70] for our privacy analysis. Compared to the ordinary DP in Definition 1, RDP is a variant of DP with a tighter bound of privacy loss.

Definition 2: A randomized algorithm $\mathcal{M}$ is $(\alpha, \epsilon(\alpha))$-RDP with $\alpha > 1$ if for any neighboring datasets $\mathcal{D}$ and $\mathcal{D}'$,

$$D_\alpha(\mathcal{M}(\mathcal{D})||\mathcal{M}(\mathcal{D}')) = \frac{1}{\alpha - 1} \log \mathbb{E}_{x \sim \mathcal{M}(\mathcal{D}')} \left[\left(\frac{\Pr[(\mathcal{D}) = x]}{\Pr[(\mathcal{D}') = x]} \right)^{\alpha - 1} \right] \leq \epsilon(\alpha), \tag{11}$$

where D_α is the Rényi divergence of order α.

Before proving our main result, we describe some necessary properties of RDP in Theorems 2∼5.

Theorem 2 (Gaussian Mechanism on RDP [70], [9]): If a function f has ℓ_2-sensitivity u, then $G_\sigma \circ f$ obeys $(\alpha, \varepsilon(\alpha))$-RDP, where $\varepsilon(\alpha) = \alpha u^2/(2\sigma^2)$ and G_σ is the Gaussian mechanism defined in Section III.

Theorem 3 (Sequential Composition on RDP [70]): If the mechanism $\mathcal{M}_1$ satisfies (α, ϵ_1)-RDP and the mechanism $\mathcal{M}_2$ satisfies (α, ϵ_2)-RDP, then $\mathcal{M}_2 \circ \mathcal{M}_1$ satisfies $(\alpha, \epsilon_1 + \epsilon_2)$-RDP.

Theorem 4 (Privacy Amplification by Subsampling [71]): Let $\mathcal{M} \circ$ subsample be a randomized mechanism that first performs the subsampling without replacement with the subsampling rate γ on the dataset X and then takes as an input from the subsampled dataset X^γ. For all integers $\alpha \geq 2$, if $\mathcal{M}$ obeys $(\alpha, \epsilon(\alpha))$-RDP through Gaussian mechanism, then $\mathcal{M} \circ$ subsample satisfies $(\alpha, \epsilon'(\alpha))$-RDP, where

$$\epsilon'(\alpha) \leq \frac{1}{(\alpha - 1)} \log(1 + \gamma^2 \binom{\alpha}{2} \min\{4(e^{\epsilon(2)} - 1), e^{\epsilon(2)} \min\{2, (e^{\epsilon(\infty)} - 1)^2\}\} + \sum_{j=3}^{\alpha} \gamma^j \binom{\alpha}{j} e^{(j-1)\epsilon(j)} \min\{2, (e^{\epsilon(\infty)} - 1)^j\}), \tag{12}$$

and $\varepsilon(\alpha) = \alpha u^2/(2\sigma^2)$ with u as the sensitivity.

In the following, we use $\epsilon'(\alpha, \gamma, u)$ to indicate $\epsilon'(\alpha)$ with the subsampling rate γ and sensitivity u.

Theorem 5 (From RDP to DP [70]): If a mechanism $\mathcal{M}$ is $(\alpha, \epsilon(\alpha))$-RDP, $\mathcal{M}$ is $(\epsilon(\alpha) + \frac{\log 1/\delta}{\alpha-1}, \delta)$-DP for any $\delta \in (0, 1)$.

From Corollary 1, we know that if conv1 in C is trained by DPSGD and conv2, conv3, and FC are discarded, using conv1 in C as conv1 in D does not leak privacy. Hence, based on the above result, Theorem 6 shows the DP of DPAF.

Theorem 6: DPAF guarantees $(T_1\epsilon'(\alpha, \gamma_1, u_1) + T_2\epsilon'(\alpha, \gamma_2, u_2) + T_3\epsilon'(\alpha, \gamma_3, \sqrt{m}p) + \frac{\log \frac{1}{\delta}}{\alpha-1}, \delta)$-DP for all $\alpha \geq 2$ and $\delta \in (0, 1)$.

Proof 3: For C, our goal is to ensure that the update of conv1 is satisfied with $(\alpha, \epsilon(\alpha))$-RDP for each iteration. Note that because conv2, conv3, and FC will be discarded after the training, they do not need a DP guarantee. Let the total number of iterations for training C be T_1. Then, the DPSGD (through the Gaussian mechanism in Theorem 2) on conv1 is $(\alpha, T_1\epsilon(\alpha))$-RDP according to Theorem 3. However, it can be re-estimated as $(\alpha, T_1\epsilon'(\alpha, \gamma_1, u_1))$-RDP with subsampling rate γ_1 according to Theorem 4.

For D, since conv1's parameters are frozen during the training of D, the output of conv1 in D is satisfied with $(\alpha, T_1\epsilon'(\alpha, \gamma_1, u_1))$-RDP, because the update of conv1 in C has been proven to be RDP. Let the total number of iterations for training conv2* be T_2. Then, the DPSGD (through the Gaussian mechanism in Theorem 2) on conv2* is $(\alpha, T_2\epsilon(\alpha))$-RDP according to and Theorem 3. Similarly, the update of conv2* is satisfied with $(\alpha, T_2\epsilon'(\alpha, \gamma_2, u_2))$-RDP with subsampling rate γ_2 according to Theorem 4. So far, the joint consideration of conv1 and conv2*, (conv1, conv2*), is satisfied with $(\alpha, T_1\epsilon'(\alpha, \gamma_1, u_1) + T_2\epsilon'(\alpha, \gamma_2, u_2))$-RDP guarantee according to Theorem 3. Unlike the cases of conv1 and conv2*, where noise is injected in the backward phase, the noise injection to AGG occurs in the forward phase. More specifically, we set a DPAGG to aggregate input data and add noise in the aggregated data. The sensitivity of AGG has been calculated as $\sqrt{m}p$ in Eq. (4). Let the number of iterations for the training of D be T_3. The DPAGG with the noise sampled from $N(0, mp^2\sigma^2)$ is satisfied with $(\alpha, T_3\epsilon'(\alpha, \gamma_3, \sqrt{m}p))$-RDP with subsampling ratio γ_3 according to Theorem 2 and Theorem 3. Thus, the joint consideration of conv1, conv2*, and DPAGG, (conv1, conv2*, DPAGG), will fulfill $(\alpha, T_1\epsilon'(\alpha, \gamma_1, u_1) + T_2\epsilon'(\alpha, \gamma_2, u_2) + T_3\epsilon'(\alpha, \gamma_3, \sqrt{m}p))$-RDP according to Theorem 3. Because the DPAGG has DP guarantee, the update of conv3* and FC* is satisfied with RDP by the postprocessing. Finally, the update of G does not access the sensitive data and, as a result, is satisfied with RDP by the postprocessing. Overall, according to Theorem 5, DPAF is satisfied with $(T_1\epsilon'(\alpha, \gamma_1, u_1) + T_2\epsilon'(\alpha, \gamma_2, u_2) + T_3\epsilon'(\alpha, \gamma_3, \sqrt{m}p) + \frac{\log \frac{1}{\delta}}{\alpha-1}, \delta)$-DP.