

## Supplementary Material

### Procedure of PGE

The procedure of our approach is shown in Alg. 1 and Alg. 2.

Alg. 1 is the training procedure of the network. It can be described as the following four steps:

- randomly sample a batch of data :  $\{\mathbf{x}_i\}_{i=1}^b$ .
- randomly sample a batch of vectors  $\mathbf{v} = \{\mathbf{v}_i\}_{i=1}^b$ .
- form  $\mathbf{v}^-$  and compute  $\mathbf{v}^r = \mathcal{R}(\mathbf{v}|\mathbf{v}^-)$ .
- calculate the loss function with Eq. (8) and update the network.

We note that the network is trained to predict the direction where the logarithmic data density grows the most. We are more interested in the direction of the predictions than the scale, so we choose cosine distance to select the top- $k$  nearest projection vectors to form  $\mathbf{v}^-$ .

Alg. 2 is the sampling procedure of generated images. Unlike traditional Hamiltonian dynamics, we adjust the step size  $\lambda$  every fixed number of epochs (line. 4 in Alg. 2). This enables faster sampling and does not get trapped in a local optimum. Every  $N$  rounds of sampling, we perform an acceptance-rejection strategy to improve the fidelity and diversity of the generated images (line. 10-11 in Alg. 2).

---

#### Algorithm 1 Private Gradient Estimation

**Input:** Private data  $\mathbf{x}$ , training iterations  $T$ , loss function  $\mathcal{L}(\theta; \cdot)$ , DP mechanism  $\mathcal{R}(\cdot)$ , randomized response parameter  $k$ , learning rate  $\gamma$

- 1: Initialize  $\theta_0$  randomly
  - 2: **for**  $t \in [T]$  **do**
  - 3:   Sample a batch of data  $\{\mathbf{x}_i\}_{i=1}^b$  from  $\mathbf{x}$
  - 4:   Sample a batch of vectors  $\{\mathbf{v}_i\}_{i=1}^b$  from a Gaussian distribution
  - 5:   Calculate loss  $\mathcal{L}(\theta; \{\mathbf{x}_i\}_{i=1}^b, \{\mathbf{v}_i\}_{i=1}^b, \mathcal{R}(\cdot))$
  - 6:   Update the network  $\theta_{t+1} \leftarrow \theta_t + \gamma \nabla \mathcal{L}$
  - 7: **end for**
  - 8: **return**  $\theta_T$
  - 9: **Function**  $\mathcal{R}(\mathbf{v})$
  - 10: Initialize  $\mathbf{v}_r$  to be an empty set  $\Phi$
  - 11: **for** each  $\mathbf{v}_i$  in  $\mathbf{v}$  **do**
  - 12:   Select top- $k$  nearest vectors to  $\mathbf{v}_i$  from  $\mathbf{v}$  to form  $\mathbf{v}^-$
  - 13:   Append  $\mathbf{v}_i$  to  $\mathbf{v}_r$  with probability of  $\frac{e^\epsilon}{e^\epsilon + k - 1}$  and append other elements in  $\mathbf{v}^-$  with probability of  $\frac{1}{e^\epsilon + k - 1}$
  - 14: **end for**
  - 15: **return**  $\mathbf{v}_r$
- 

---

#### Algorithm 2 MCMC Sampling with Hamiltonian Dynamics

**Input:** Trained network  $q_\theta(\cdot)$ , kinetic energy  $K(\cdot)$ , step size  $\lambda_0$ , private data  $D$ , sampling iterations  $M$ , acceptance-rejection iterations  $N$

- 1: Initialize  $\mathbf{x}(0), \mathbf{c}(0)$  randomly
  - 2: **for**  $m \in [M]$  **do**
  - 3:   Initialize  $p(m)$  randomly
  - 4:    $\lambda = \lambda_0 \cdot (M/m)^2$
  - 5:   **for**  $n \in [N]$  **do**
  - 6:      $\mathbf{c} \left( m + (n + \frac{1}{2})\lambda \right) = \mathbf{c}(m + n\lambda) - \frac{\lambda}{2} q_\theta(\mathbf{x}(m + n\lambda))$
  - 7:      $\mathbf{x}(m + (n + 1)\lambda) = \mathbf{x}(m + n\lambda) + \lambda \nabla_{\mathbf{c}} K(\mathbf{c} \left( m + (n + \frac{1}{2})\lambda \right))$
  - 8:      $\mathbf{c}(m + (n + 1)\lambda) = \mathbf{c} \left( m + (n + \frac{1}{2})\lambda \right) - \frac{\lambda}{2} q_\theta(\mathbf{x}(m + (n + 1)\lambda))$
  - 9:   **end for**
  - 10:   Calculate the probability  $p_a = \min(1, q_\theta(\mathbf{x}(m + N\lambda))/q_\theta(\mathbf{x}(m)))$
  - 11:    $\mathbf{x}(m + 1) = \mathbf{x}(m + N\lambda)$  with probability  $p_a$  and  $\mathbf{x}(m + 1) = \mathbf{x}(m)$  with  $1 - p_a$
  - 12: **end for**
  - 13: **return**  $\mathbf{x}(M)$
-

### Proof of Theorem. 1

Recall the core thought of our PGE, we perturb the projection vector of log data density to achieve differential privacy. We aim to protect the log data density  $(\mathbf{v}^T)^* \mathbf{v}^T \nabla \log p(\mathbf{x})$ , which guides the network learning.  $\mathbf{v}^*$  represents the inverse matrix of  $\mathbf{v}$ . Given a batch of data  $D = \{\mathbf{x}_i\}_{i=1}^b$  and projection vector  $\mathbf{v} = \{\mathbf{v}_i\}_{i=1}^b$ , the probability of  $\mathbf{v}_i^* \mathbf{v}_i \nabla \log p(\mathbf{x}_i)$  is as follows:

$$\Pr[\mathbf{v}_i, \mathbf{x}_i | \mathbf{v}, D] = \Pr[\mathbf{x}_i | D] \cdot \Pr[\mathbf{v}_i | \mathbf{v}] \cdot \Pr[\mathbf{v}^- | \mathbf{v}_i, \mathbf{v}] \cdot \Pr[\mathcal{R}(\mathbf{v}_i) = \mathbf{v}_i^r | \mathbf{v}^-]. \quad (1)$$

After we achieve differential privacy by performing randomized response, the probability of the resulting output  $\mathbf{v}_i^* \mathbf{v}_i^r \nabla \log p(\mathbf{x}_i) = \mathbf{v}_i^* \mathcal{R}(\mathbf{v}_i) \nabla \log p(\mathbf{x}_i)$  is as follows:

$$\Pr[\mathbf{v}_i, \mathbf{x}_i, \mathbf{v}_i^r, \mathbf{v}^- | \mathbf{v}, D] = \Pr[\mathbf{x}_i | D] \cdot \Pr[\mathbf{v}_i | \mathbf{v}] \cdot \Pr[\mathbf{v}^- | \mathbf{v}_i, \mathbf{v}] \cdot \Pr[\mathcal{R}(\mathbf{v}_i) = \mathbf{v}_i^r | \mathbf{v}^-]. \quad (2)$$

This equation is obtained by Bayes' theorem. In our approach, we sample data  $\mathbf{x}_i$  and  $\mathbf{v}_i$  uniformly, so that  $\Pr[\mathbf{x}_i | D]$  and  $\Pr[\mathbf{v}_i | \mathbf{v}]$  are  $1/b$ . Given  $\mathbf{v}$  and  $\mathbf{v}_i$ , we select the top- $k$  vectors from  $\mathbf{v}$  that are similar to  $\mathbf{v}_i$  to form  $\mathbf{v}^-$ . This process is fixed, so  $\Pr[\mathbf{v}^- | \mathbf{v}_i, \mathbf{v}] = 1$ . Following the above analysis, we have,

$$\Pr[\mathbf{x}_i, \mathbf{v}_i, \mathbf{v}_i^r, \mathbf{v}^- | \mathbf{v}, D] = \Pr[\mathcal{R}(\mathbf{v}_i) = \mathbf{v}_i^r | \mathbf{v}^-] \cdot 1/b^2. \quad (3)$$

Given a image  $\mathbf{x}_i$  and its projection vector  $\mathbf{v}_i$ , we define  $\mathcal{M}(\mathbf{v}_i, \mathbf{u}_i) = \mathcal{R}(\mathbf{v}_i) \cdot \mathcal{H}(\mathbf{u}_i) = \mathbf{v}_i^r \cdot \nabla \log p(\mathbf{x}_i)$ . In our case, we assume that  $\mathcal{R}$  and  $\mathcal{H}$  are independent of each other, so  $\Pr[\mathcal{M}(\cdot)] = \Pr[\mathcal{R}(\cdot)] \cdot \Pr[\mathcal{H}(\cdot)]$ .

LEMMA 1. For any two different training data  $\mathbf{x}_i, \mathbf{x}_j$  and their projection vectors  $\mathbf{v}_i, \mathbf{v}_j$ , the mechanism  $\mathcal{M}$  satisfies

$$\Pr[\mathcal{M}(\mathbf{v}_i, \mathbf{x}_i) \in O] \leq e^\epsilon \cdot \Pr[\mathcal{M}(\mathbf{v}_j, \mathbf{x}_j) \in O], \quad (4)$$

where  $O$  is a possible output of  $\mathcal{M}$ .

PROOF. From the definition of randomized response mechanism, we can know the probability that  $\mathcal{R}(\cdot)$  takes as input  $\mathbf{v}_i$  and returns  $\mathbf{v}_i$  is the largest for  $e^\epsilon / (e^\epsilon + k - 1)$  and that takes as input  $\mathbf{v}_j$  and returns  $\mathbf{v}_i$  is the smallest for  $1 / (e^\epsilon + k - 1)$ . We sample  $\mathbf{x}_i$  uniformly, so  $\Pr[\mathcal{H}(\mathbf{x}_i)] = \Pr[\mathcal{H}(\mathbf{x}_j)]$ . Then we have

$$\begin{aligned} \Pr[\mathcal{M}(\mathbf{v}_i, \mathbf{x}_i) \in O] &= \Pr[\mathcal{R}(\mathbf{v}_i) = \mathbf{v}_i] \cdot \Pr[\mathcal{H}(\mathbf{x}_i)] \\ &\leq e^\epsilon \cdot \Pr[\mathcal{R}(\mathbf{v}_j) = \mathbf{v}_i] \cdot \Pr[\mathcal{H}(\mathbf{x}_i)] \\ &= e^\epsilon \cdot \Pr[\mathcal{R}(\mathbf{v}_j) = \mathbf{v}_i] \cdot \Pr[\mathcal{H}(\mathbf{x}_j)] \\ &= e^\epsilon \cdot \Pr[\mathcal{M}(\mathbf{v}_j, \mathbf{x}_j) \in O] \end{aligned} \quad (5)$$

□

LEMMA 2. The mechanism  $\mathcal{M}$  satisfies  $\epsilon$ -DP.

PROOF. Consider two adjacent datasets  $D = \{\mathbf{x}_i\}_{i=1}^b, D' = \{\mathbf{x}'_i\}_{i=1}^b$  that differ only by one data and their projection vectors  $\mathbf{v} = \{\mathbf{v}_i\}_{i=1}^b, \mathbf{v}' = \{\mathbf{v}'_i\}_{i=1}^b$ . The data are independent of each other so we have

$$\begin{aligned} \Pr[\mathcal{M}(\mathbf{v}, D) \in O] &= \Pr[\mathcal{M}(\mathbf{v} \cap \mathbf{v}', D \cap D') \in O] \cdot \Pr[\mathcal{M}(\mathbf{v}_i, \mathbf{x}_i) \in O] \\ &\leq e^\epsilon \cdot \Pr[\mathcal{M}(\mathbf{v} \cap \mathbf{v}', D \cap D') \in O] \cdot \Pr[\mathcal{M}(\mathbf{v}'_i, \mathbf{x}'_i) \in O] \\ &= e^\epsilon \cdot \Pr[\mathcal{M}(\mathbf{v}', D') \in O], \end{aligned} \quad (6)$$

where  $O$  is a set of possible outputs. From line 2 to line 3 is based on Lemma. 1. □

Our approach trains the network with datasets  $D$  and projection vectors  $\mathbf{v}$ . It is necessary to calculate the joint probability to clarify the association of each vector. Next, we prove that our PGE satisfies differential privacy based on Lemma. 2.

THEOREM 1. Our PGE satisfies  $\epsilon$ -DP.

PROOF. Consider two adjacent datasets  $D = \{\mathbf{x}_i\}_{i=1}^b$  and  $D' = \{\mathbf{x}'_i\}_{i=1}^b$  and their projection vectors  $\mathbf{v} = \{\mathbf{v}_i\}_{i=1}^b$ ,  $\mathbf{v}' = \{\mathbf{v}'_i\}_{i=1}^b$ . We define  $\mathcal{F}(\mathbf{v}_i, \mathbf{x}_i) = \mathbf{v}_i^* \mathcal{R}(\mathbf{v}_i) \nabla \log p(\mathbf{x}_i)$ , then according to Eq. (2) and Eq. (3), we have

$$\begin{aligned}
\Pr[\mathcal{F}(\mathbf{v}, D) \in O] &= \prod_i \Pr[\mathbf{v}_i, \mathbf{x}_i, \mathbf{v}'_i, \mathbf{v}^- | \mathbf{v}, D] \\
&= \prod_i \Pr[\mathcal{R}(\mathbf{v}_i) = \mathbf{v}'_i | \mathbf{v}^-] \cdot 1/b^2 \\
&= \prod_i \Pr[\mathcal{R}(\mathbf{v}_i) \cdot \mathcal{H}(\mathbf{x}_i) \in O | \mathbf{v}^-] \cdot 1/b^2 \\
&= \Pr[\mathcal{M}(\mathbf{v}, D) \in O] \cdot 1/b^2 \\
&\leq e^\epsilon \cdot \Pr[\mathcal{M}(\mathbf{v}', D') \in O] \cdot 1/b^2 \\
&= e^\epsilon \cdot \Pr[\mathcal{F}(\mathbf{v}', D') \in O],
\end{aligned} \tag{7}$$

where  $O$  is the range of output of  $\mathcal{F}$ , from line 4 to line 5 is based on Lemma. 2 and from line 5 to line 6 is the inverse derivation of line 1 to line 4. We note that as long as  $\mathcal{F}(\mathbf{v}, D)$  satisfies differential privacy, the trained probabilistic model and the images generated with it also satisfy differential privacy according to the post-processing property of differential privacy. So our PGE satisfies  $\epsilon$ -DP.  $\square$

### Covergence Analysis

We begin by stating that most of our analysis process is based on [Bottou et al., 2018]. We consider the worst-case scenario: the gradient of the randomized response algorithm when the outputs and inputs are different is exactly the opposite of the gradient when they are the same. In this case, the  $\mathcal{R}(\cdot)$  algorithm performing on the label is equivalent to performing on the gradient. To make the analysis easier and more understandable, we rewrite  $q_\theta(\cdot)$  as  $q(\theta; \cdot)$  and follow five standard assumptions same as [Bottou et al., 2018],

$$\begin{aligned}
(1) \quad & \|\nabla q(\theta; \cdot) - \nabla q(\theta'; \cdot)\|_2 \leq \kappa \|\theta - \theta'\|_2; \\
(2) \quad & q(\theta; \cdot) \geq q(\theta'; \cdot) + \nabla q(\theta'; \cdot)^T (\theta - \theta') + \frac{1}{2} c \|\theta - \theta'\|_2^2 \\
(3) \quad & \nabla q(\theta; \cdot)^T \mathbb{E}_{\mathbf{x}} [g(\theta; \mathbf{x})] \geq \mu \|\nabla q(\theta; \cdot)\|_2^2; \\
(4) \quad & \|\mathbb{E}_{\mathbf{x}} [g(\theta; \cdot)]\|_2 \leq \mu_G \|\nabla q(\theta; \cdot)\|_2; \\
(5) \quad & \mathbb{V}_{\mathbf{x}} [g(\theta; \cdot)] \leq C + \mu_V \|\nabla q(\theta; \cdot)\|_2^2,
\end{aligned} \tag{8}$$

where  $\theta$  and  $\theta'$  are the weights of model  $q$ ,  $\nabla q(\theta; \cdot)$  is the true gradient,  $g(\theta; \cdot)$  is the gradient we computed,  $\mathbb{E}[\cdot]$  is the symbol for mean calculation,  $\mathbb{V}[\cdot]$  is the symbol for variance calculation and  $\kappa, c, \mu, \mu_G, \mu_V, C$  are non-negative constants.

LEMMA 3. For any two weights  $\theta$  and  $\theta'$ , the difference of the objective function  $q(\theta) - q(\theta')$  is limited by the distance between the weights.

$$q(\theta; \cdot) \leq q(\theta'; \cdot) + \nabla q(\theta'; \cdot)^T (\theta - \theta') + \frac{1}{2} \kappa \|\theta - \theta'\|_2^2. \tag{9}$$

PROOF. Consider any path  $s$  from  $\theta'$  to  $\theta$ , we have

$$\begin{aligned}
q(\theta; \cdot) - q(\theta'; \cdot) &= \int_s \nabla q(\mathbf{x}; \cdot)^T d\mathbf{x} \\
&= \int_0^1 \frac{\partial q(s(t); \cdot)}{\partial t} dt \\
&= \int_{\theta'}^\theta \nabla q(s(t); \cdot) ds(t) \\
&= \int_{\theta'}^\theta \nabla q(\theta'; \cdot) ds(t) + \int_{\theta'}^\theta [\nabla q(s(t); \cdot) - \nabla q(\theta'; \cdot)] ds(t) \\
&\leq \int_{\theta'}^\theta \nabla q(\theta'; \cdot) ds(t) + \int_{\theta'}^\theta \kappa \|s(t) - \theta'\|_2 ds(t) \\
&= \nabla q(\theta'; \cdot)^T (\theta - \theta') + \frac{1}{2} \kappa \|\theta - \theta'\|_2^2.
\end{aligned} \tag{10}$$

$\square$

The inequality is based on assumption (1).

LEMMA 4. For any weight  $\theta$ , the distance between  $q(\theta; \cdot)$  and the minimum value  $q(\theta^*; \cdot)$  is limited by  $\nabla q(\theta; \cdot)$  as follows

$$q(\theta; \cdot) - q(\theta^*; \cdot) \leq \frac{1}{2c} \|\nabla q(\theta; \cdot)\|_2^2. \tag{11}$$

PROOF. According to assumption (2), we can regard the right side of the inequality as a quadratic function on  $\theta$ . When  $\theta = \theta' - \frac{1}{c}\nabla q(\theta'; \cdot)$ , it takes the minimum value  $q(\theta'; \cdot) - \frac{1}{2c}\|\nabla q(\theta'; \cdot)\|_2^2$ . Substituting it into assumption (2) and letting  $\theta = \theta^*$ , we can get Lemma 4.  $\square$

According to the assumptions before (We consider the worst-case scenario: the gradient of the randomized response algorithm when the outputs and inputs are different is exactly the opposite of the gradient when they are the same.), we consider the update at step  $k$  as

$$\theta_{k+1} = \theta_k - \gamma \cdot \mathcal{R}(g(\theta_k, \cdot)), \quad (12)$$

where  $\mathcal{R}(g(\theta_k, \cdot))$  will return  $g(\theta_k, \cdot)$  with the probability of  $e^\varepsilon/(e^\varepsilon + k - 1)$  and return  $-g(\theta_k, \cdot)$  with the probability of  $1 - e^\varepsilon/(e^\varepsilon + k - 1)$ .

Based on Lemma 3, we have

$$q(\theta_{k+1}; \cdot) \leq q(\theta_k; \cdot) - \gamma \nabla q(\theta_k; \cdot)^T \mathcal{R}(g(\theta_k, \cdot)) + \frac{1}{2} \kappa \gamma^2 \frac{\|\mathcal{R}(g(\theta_k, \cdot))\|_2^2}{\|g(\theta_k, \cdot)\|_2^2}. \quad (13)$$

Taking the expectations on both sides gives

$$\begin{aligned} \mathbb{E}[q(\theta_{k+1}; \cdot) - q(\theta_k; \cdot)] &\leq -\gamma \nabla q(\theta_k; \cdot)^T \mathbb{E}[\mathcal{R}(g(\theta_k, \cdot))] \\ &\quad + \frac{1}{2} \gamma^2 \kappa \frac{\mathbb{E}[\|g(\theta_k, \cdot)\|_2^2]}{\|\mathbb{E}[g(\theta_k, \cdot)]\|_2^2 + \mathbb{V}[g(\theta_k, \cdot)]}. \end{aligned} \quad (14)$$

According to our pre-assumed scenario,

$$\begin{aligned} \mathbb{E}[\mathcal{R}(g(\theta_k, \cdot))] &= \frac{e^\varepsilon}{e^\varepsilon + k - 1} g(\theta_k, \cdot) \\ &\quad + \left(1 - \frac{e^\varepsilon}{e^\varepsilon + k - 1}\right) (-g(\theta_k, \cdot)) \\ &= \underbrace{\left(\frac{2e^\varepsilon}{e^\varepsilon + k - 1} - 1\right)}_{\zeta} g(\theta_k, \cdot). \end{aligned} \quad (15)$$

Combined with assumptions (3), (4) and (5), we can get

$$\begin{aligned} \mathbb{E}[q(\theta_{k+1}; \cdot) - q(\theta_k; \cdot)] &\leq -\gamma \zeta \nabla q(\theta_k; \cdot)^T \mathbb{E}[g(\theta_k, \cdot)] + \frac{1}{2} \gamma^2 \kappa (\|\mathbb{E}[g(\theta_k, \cdot)]\|_2^2 + \mathbb{V}[g(\theta_k, \cdot)]) \\ &\leq -\gamma \zeta \mu \|\nabla q(\theta_k; \cdot)\|_2^2 + \frac{1}{2} \gamma^2 \kappa (C + (\mu_G^2 + \mu_V) \|q(\theta_k; \cdot)\|_2^2) \\ &= \underbrace{(-\gamma \zeta \mu + \frac{1}{2} \gamma^2 \kappa (\mu_G^2 + \mu_V))}_{\tau} \|q(\theta_k; \cdot)\|_2^2 + \frac{1}{2} \gamma^2 \kappa C. \end{aligned} \quad (16)$$

If the algorithm converges, it takes  $-\gamma \mu + \frac{1}{2} \gamma^2 \kappa (\mu_G^2 + \mu_V) < 0$ . According to Lemma 4, we can further get

$$\begin{aligned} \mathbb{E}[q(\theta_{k+1}; \cdot) - q(\theta^*; \cdot)] - \mathbb{E}[q(\theta_k; \cdot) - q(\theta^*; \cdot)] &\leq \tau \|q(\theta_k; \cdot)\|_2^2 + \frac{1}{2} \gamma^2 \kappa C \\ &\leq 2\tau c \mathbb{E}[q(\theta_k; \cdot) - q(\theta^*; \cdot)] + \frac{1}{2} \gamma^2 \kappa C. \end{aligned} \quad (17)$$

Eq. 17 is transformed to obtain

$$\mathbb{E}[q(\theta_{k+1}; \cdot) - q(\theta^*; \cdot)] + \frac{\gamma^2 \kappa C}{4\tau c} \leq (2\tau c + 1) (\mathbb{E}[q(\theta_k; \cdot) - q(\theta^*; \cdot)] + \frac{\gamma^2 \kappa C}{4\tau c}) \quad (18)$$

We know  $\tau < 0$ , so  $2\tau c + 1 < 1$ . The algorithm converges when we guarantee that  $\tau < 0$ . The error from the minimum  $q(\theta^*; \cdot)$  is  $-\frac{\gamma^2 \kappa C}{4\tau c}$ .

## Experimental Results under Small $\varepsilon$

We conduct experiments under small  $\varepsilon$  to verify the effectiveness of our method here. The results are shown in Tab. 1. We can find that even under the condition of small  $\varepsilon$ , our method still has outstanding performance.

## Discussion of Residual Structure

We use a residual structure in our framework. Here, we conduct experiments on two datasets (MNIST and FMNIST) to explore the role of this structure. The results are shown in Tab. 2. We capture the diversity of the generated samples through the metric of entropy. We find that having this structure improves the diversity and data utility of the generated samples.

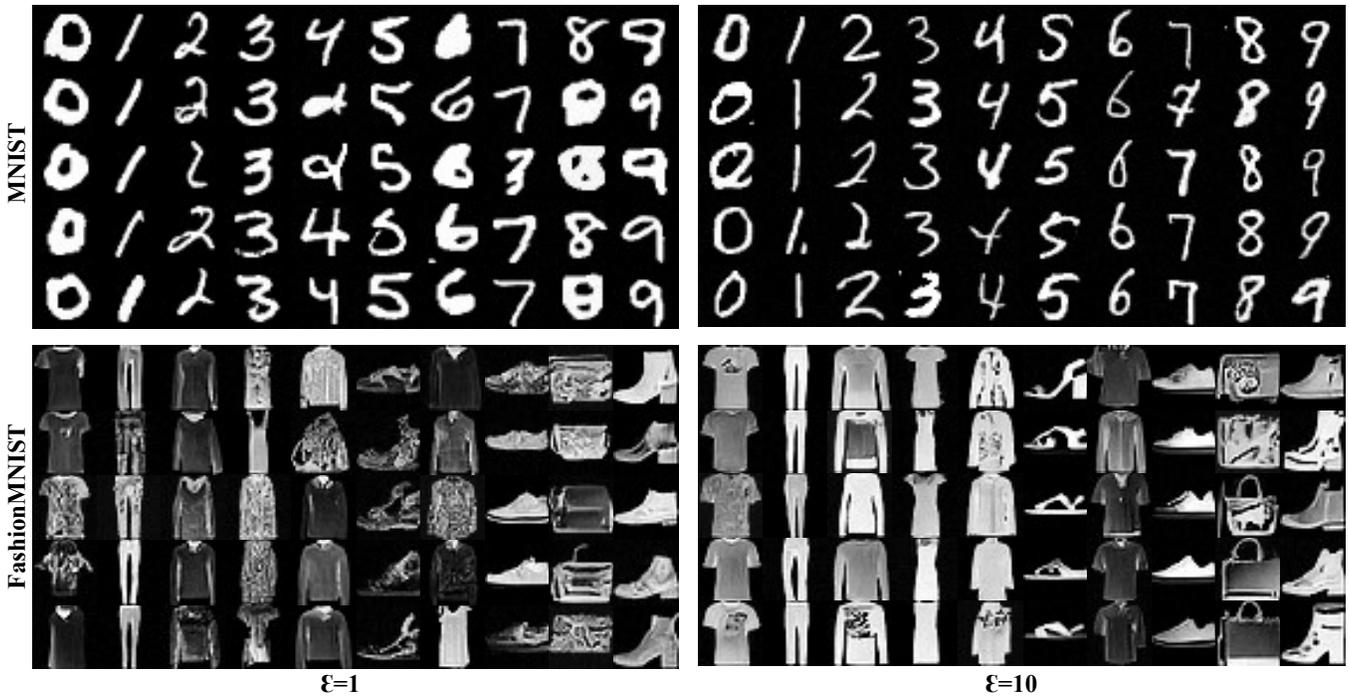
**Table 1: Classification accuracy comparisons on image datasets under small privacy budget  $\epsilon$ .**

$\epsilon$	MNIST			FMNIST		
	0.2	0.6	0.8	0.2	0.6	0.8
DataLens	0.2344	0.4201	0.6485	0.2226	0.3863	0.5534
PGE	<b>0.4702</b>	<b>0.7462</b>	<b>0.9211</b>	<b>0.4582</b>	<b>0.6954</b>	<b>0.8002</b>

**Table 2: Classification accuracy and entropy comparisons on image datasets under small privacy budget  $\epsilon$ .**

Acc./Entropy	MNIST	FMNIST
With Res.	<b>0.9751/3.21</b>	<b>0.8934/3.23</b>
Without Res.	0.9543/3.14	0.8761/3.11

**Extended Visualization Results**



**Figure 1: Visualization results of MNIST and FashionMNIST with 28x28 resolution under different privacy budget.**

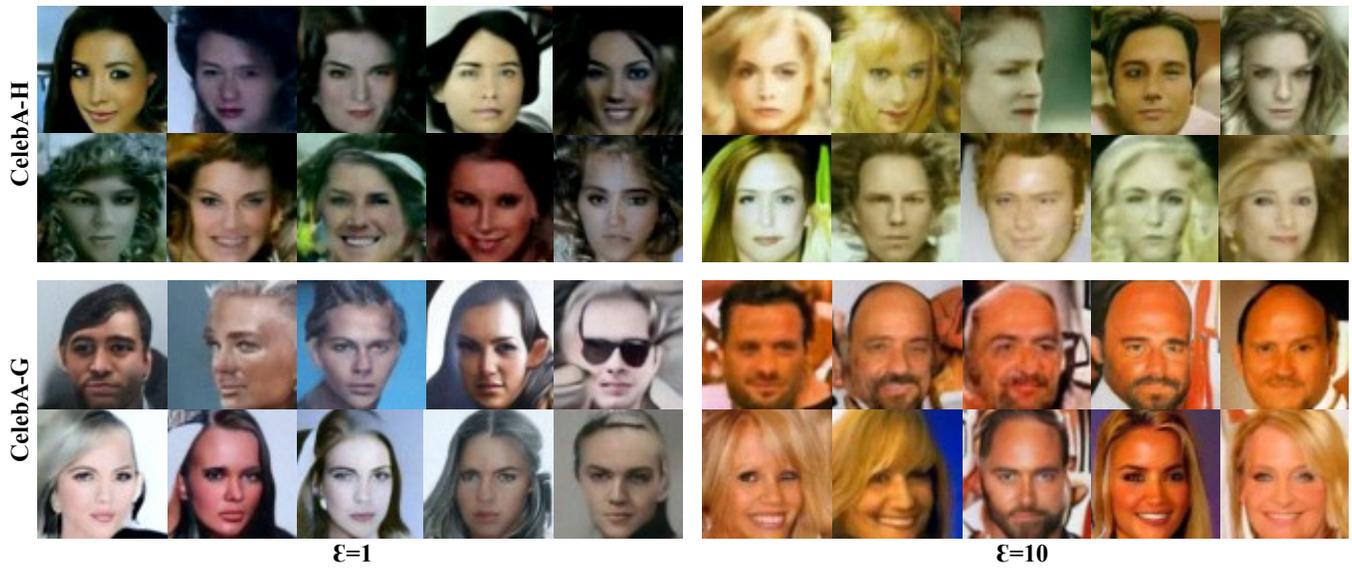


Figure 2: Visualization results of CelebA with 64×64 resolution under different privacy budget.