DPWGAN: High-Quality Load Profiles Synthesis With Differential Privacy Guarantees

Jiaqi Huang , Qiushi Huang, Gaoyang Mou, and Chenye Wu

INTRODUCTION

- Smart meters gather detailed load data for smart grid efficiency. Yet, privacy concerns arise from potential data leaks. Current solutions use perturbation or generation mechanisms for privacy in load profile analysis, but they're either inflexible or not entirely effective in preventing leakage.
- Global smart meter installations expected to reach 198.53 million units by 2026. They enable data analytics for load forecasting, behavior analysis, and demand-side response. However, privacy concerns about revealing user habits and appliance ownership reduce willingness to install and share load profiles.

Main Objective

- To this end, we propose a dif ferentially private Wasserstein Generative Adversarial Networks (DPWGAN) approach in this study. This approach can privately convert a real-world dataset into a high-quality synthetic load dataset so that studies and analyses conducted on the synthetic dataset can automatically satisfy user-level differential privacy guarantees. The extensive numerical studies highlight that our approach acts as an excellent substitute for the original dataset in real-world load profiling tasks.

Differenetial privacy

- Differential privacy is a mathematical way to protect individuals when their data is used in data sets. It ensures that an individual will experience no difference whether they participate in information collection or not. This means that no harm will come to the participant as a result of providing data.
- The mathematical equation of differential privacy is

$$\left|\log\left(\frac{\Pr[\mathcal{M}(D) = s]}{\Pr[\mathcal{M}(D') = s]}\right)\right| \leq \epsilon,$$

- A randomized mechanism M : D → R with domain D and range R satis f ies (,δ)-differential privacy if for every pair of neighboring datasets D,D∈ D and for every point s in output range R

Differentially Private Stochastic Gradient Descent and Moments Accountant

- Differentially private stochastic gradient descent (DPSGD) is a method for private learning by adding noise to gradients during training. It ensures differential privacy by randomly selecting batches of training samples, clipping their gradients, and aggregating them with noisy sums. This process is repeated at each training step, with the privacy cost calculated by composing the privacy consumption at each step. However, the composition of (ε, δ)-differential privacy can be computationally complex and prone to privacy waste, whereas the composition of RDP is simpler and tighter, without wasting the privacy budget.
- Privacy Loss: For a randomized mechanism Mand D,D a pair of neighboring datasets. Let aux denote an auxiliary input. For an outcome o ∈ R, the privacy loss at o is defined as:

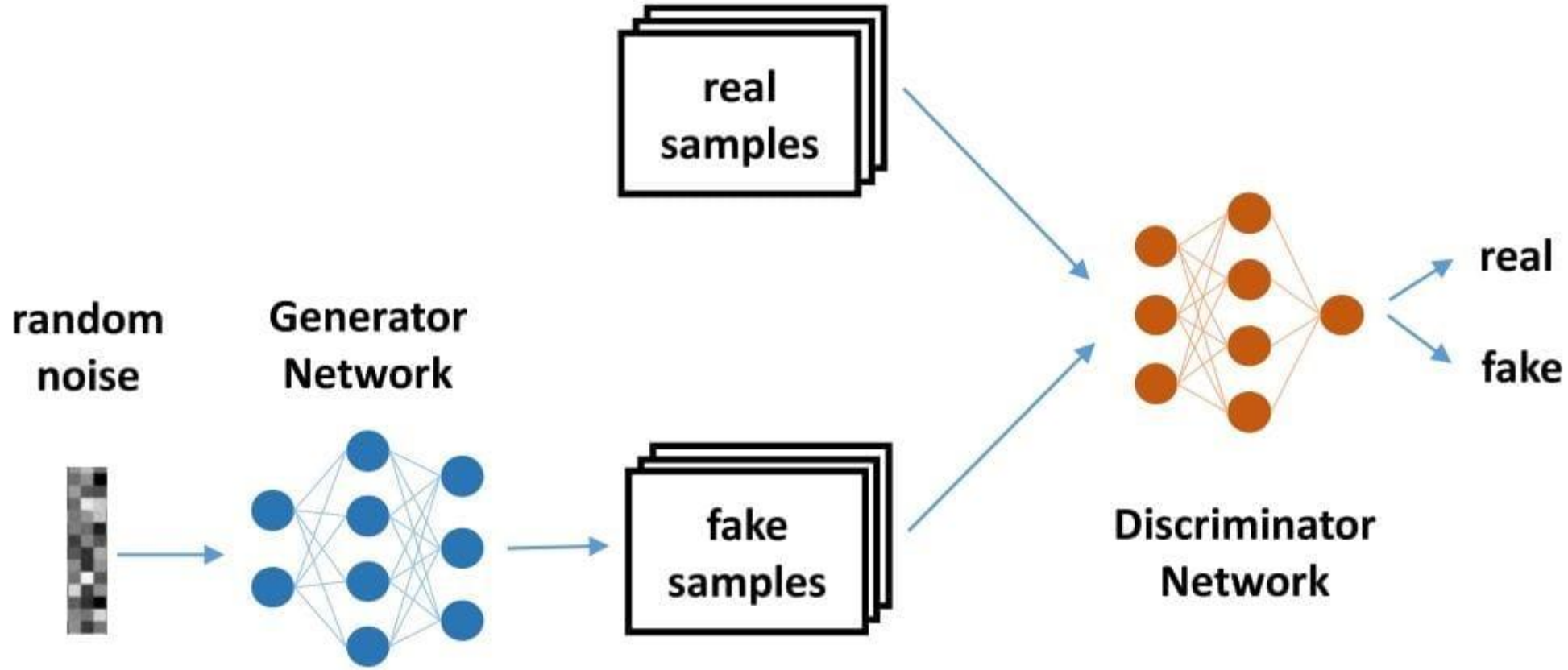
$$c(o; \mathcal{M}, aux, D, D') \triangleq \log \frac{\Pr[\mathcal{M}(aux, D) = o]}{\Pr[\mathcal{M}(aux, D') = o]}.$$

GAN and WGAN

- The Generator Network (GN) and the Discriminator Network (DN) are the main components of a GAN, as shown in Figure in next slide.LetG and D be the functions representing GN and DN. The training process of GANs can be viewed as the following two-player zero-sum game with value function V(G,D):

$$\min_G \max_D V(G, D) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))].$$

- This function mathematically approximates the Jensen Shannon (JS) divergence


real
samples
random
noise
Generator
Network
fake
samples
real
fake
Discriminator
Network

However, a JS divergence based loss function suffers from the mode collapse issue, and its convergence rate is unsatisfactory in practice, increasing the privacy cost

To address these issues
we employ the Wasserstein Generative Adversarial Networks (WGAN), a widely used variant of GAN. Instead of JS divergence, WGANs play the mini max game on an approximate Wasserstein distance (i.e., Earth Mover distance):

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[D(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[D(G(\mathbf{z}))], \qquad (6)$$

DPWGAN DESIGN

- Model architecture:
- To ensure user-level differential privacy for the load dataset, each user's load profile is treated as a sample in the training dataset, requiring a sample-level differentially private deep learning algorithm. The training process of DPWGAN is considered as a randomized mechanism, with the key goal of making the parameters of DN (discriminator network) differentially private. Since GN (generator network) relies solely on DN's parameters, it inherits the same privacy guarantee. To achieve this, the DPSGD algorithm is employed to train a private DN. During training, DN and GN are alternately updated, with DN's gradients perturbed by Gaussian noise to obscure sensitive information while allowing useful statistical data features to be learned. This noise injection ensures privacy for the entire training process, as gradients are the only channel through which training samples convey information to the network.

Algorithm 1 DPWGAN

Require: Privacy target ϵ_0 and δ_0, load dataset D, weights bound of WGAN c_p, batch size n, noise scale σ, learning rate of Discriminator α_d, learning rate of Generator α_g, gradient norm bound C.

Ensure: Differentially private WGAN generator θ.

Initialize Discriminator parameters $w^{(0)}$ and Generator parameters $\theta^{(0)}$, set step number $t = 0$, privacy cost $\hat{\epsilon} = 0$, set moments accountant $\alpha(l) = 0$ for $l = 1, \ldots, L$.

while $\hat{\epsilon} < \epsilon_0$ **do**

  $t \leftarrow t + 1$

  Discriminator Step:

  Generate synthetic samples $\{\mathbf{u}_1, \ldots, \mathbf{u}_n\}$ using Generator $\theta^{(t-1)}$.

  Sample $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \overset{\text{i.i.d.}}{\sim} D$ from the load dataset.

  For each i, $\mathbf{g}_w(\mathbf{x}_i) \leftarrow \nabla_{w^{(t)}}(f_{w^{(t-1)}}(\mathbf{x}_i))$.

  For each i, $\mathbf{g}_w(\mathbf{u}_i) \leftarrow \nabla_{w^{(t)}}(f_{w^{(t-1)}}(\mathbf{u}_i))$.

  For each i, $\overline{\mathbf{g}}_w(\mathbf{x}_i) \leftarrow \mathbf{g}_w(\mathbf{x}_i) / \max\left(1, \frac{\|\mathbf{g}_w(\mathbf{x}_i)\|_2}{C}\right)$.

  ▷ Clip gradients.

  $\mathbf{g}_w \leftarrow \frac{1}{n}\left(\sum_{i=1}^{n}(\overline{\mathbf{g}}_w(\mathbf{x}_i) - \mathbf{g}_w(\mathbf{u}_i)) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})\right)$. ▷ Noisy aggregation of gradients.

  Update the moments accountant: $\alpha(l) = \alpha(l) + \log\left(\sum_{k=0}^{l+1}\binom{l+1}{k}(1-q)^{l+1-k}q^k \exp\left(\frac{k^2-k}{2\sigma^2}\right)\right)$ for $l = 1, \ldots, L$, where $q = \frac{n}{|D|}$ is the sampling rate.

  $\hat{\epsilon} \leftarrow \min_l \frac{\alpha(l) - log(\delta_0)}{l}$.

  Update σ according to the adaptive-noise method.

  $w^{(t)} \leftarrow w^{(t-1)} + \alpha_d \cdot RMSProp(w^{(t)}, \mathbf{g}_w)$.

  $w^{(t)} \leftarrow \text{clip}\left(w^{(t)}, -c_p, c_p\right)$

  ▷ Weights Clip.

  Generator Step:

  Generate synthetic samples $\{\mathbf{u}_1, \ldots, \mathbf{u}_n\}$ using Generator $\theta^{(t-1)}$.

  $\mathbf{g}_\theta \leftarrow -\nabla_{\theta^{(t-1)}} \frac{1}{n}\sum_{i=1}^{n} f_{w^{(t)}}(\mathbf{u}_i)$.

  $\theta^{(t)} \leftarrow \theta^{(t-1)} + \alpha_g \cdot RMSProp(\theta^{(t-1)}, \mathbf{g}_\theta)$.

end while

return $\theta^{(t-1)}$.

The Adaptive-Noise Method

- The Adaptive-Noise Method adjusts the noise scale parameter (σ) in DPSGD during the training process. Initially, a larger σ is used to allow quick progress towards the optimization goal, saving privacy cost. As training progresses and the model nears convergence, σ is gradually reduced to provide more precise gradients for fine-tuning. This strategy, shown to be beneficial in previous studies, is crucial for stable and effective training in DPWGAN. In DPWGAN, a simple exponential decay strategy is employed to adjust the noise scale after each training step, ensuring stable and effective training.

Clustering-Divergence for Synthetic Load Profile Evaluation

- Evaluating the quality of data generated by GANs, particularly for time-series data like load profiles, presents challenges due to the absence of established evaluation methods. Unlike synthetic image evaluation methods such as the Inception score, there's a lack of research on evaluating generated time-series data. To address this, a suitable indicator for evaluating load profile synthesis quality is needed.

- The ultimate goal is to assess whether common power consumption patterns are preserved in the synthetic load dataset. However, distinguishing between common and unique personal patterns is difficult, and required patterns vary depending on the task. To comprehensively evaluate, the focus shifts to the statistical distribution of consumption, as it's challenging to directly depict this continuous random variable through finite samples. Load clustering, a temporal assessment tool, is employed to reveal insights into residents' electricity usage patterns. K-means clustering is commonly used for this purpose, measuring similarity between load series based on Euclidean distance. Users in the same cluster likely share similar living habits, housing, and family status. Clustering centers represent users within clusters, and the frequency of samples in each cluster approximates the dataset's data distribution. This approach enables comparison of different datasets based on the probability distribution of samples falling into each cluster.

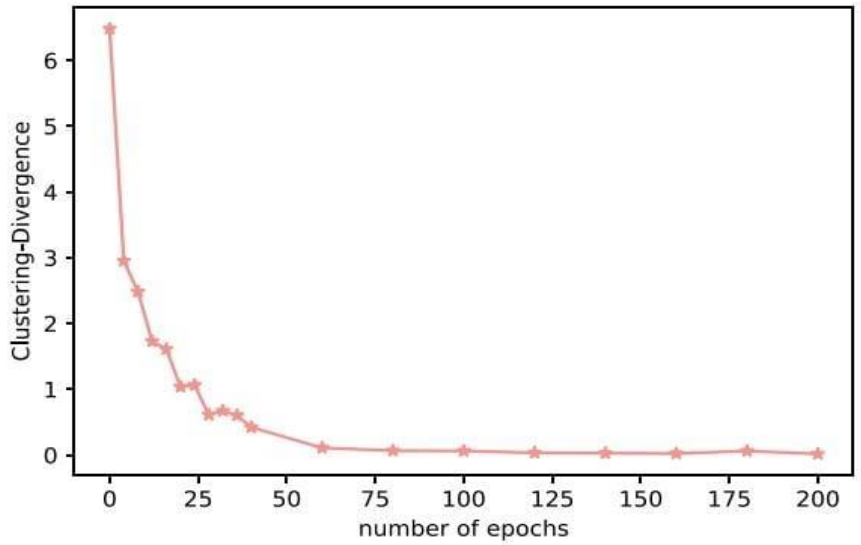
- To calculate the Clustering-Divergence, we first conduct a K-means clustering on the original dataset with an appropriate number of clusters. The resulting clustering centers catego rize all the users, and the users' frequency in each cluster is recorded as the distribution of the original dataset, denoted as

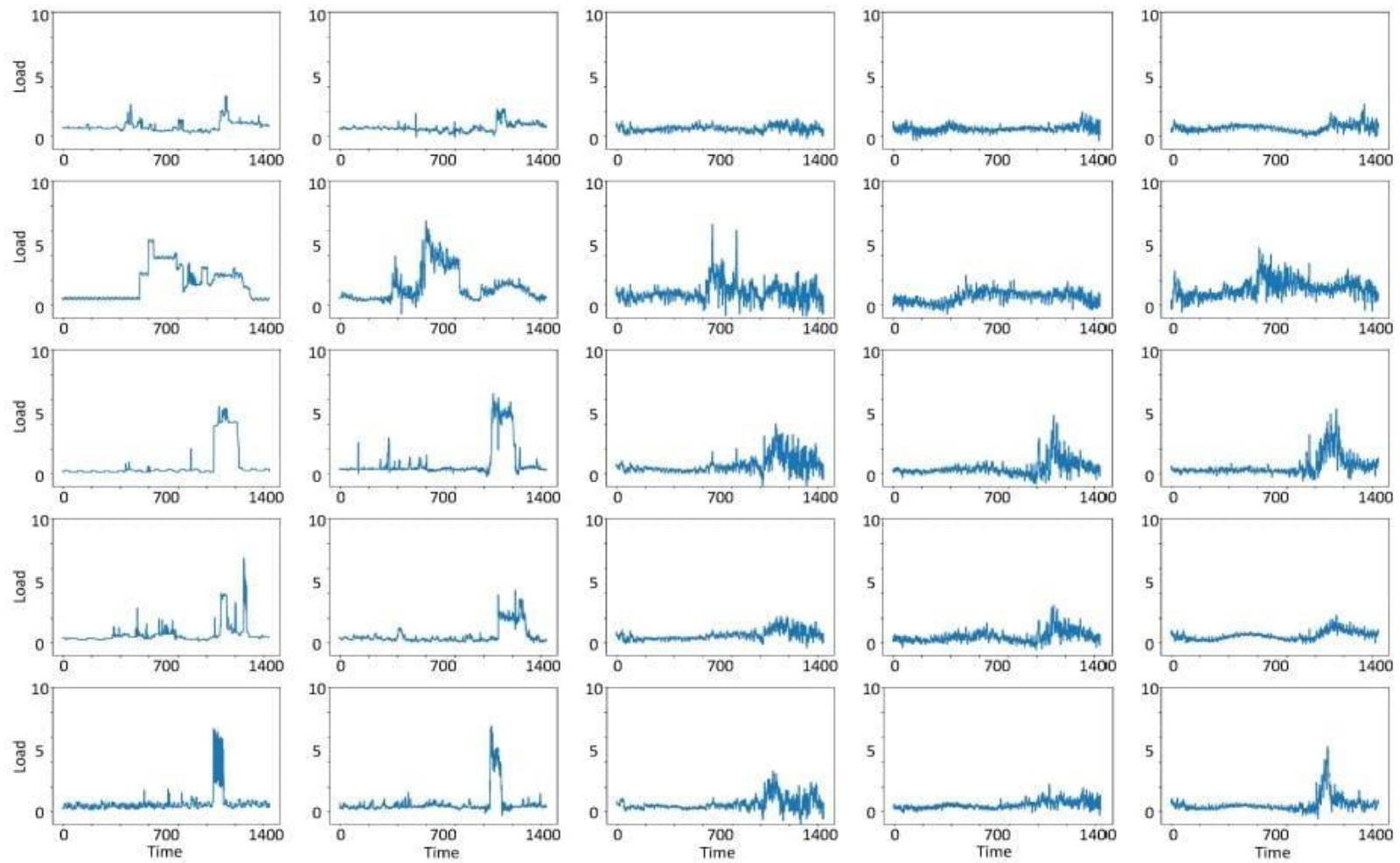
$$P_o = \{f_1, f_2, \ldots, f_k\},$$

- where k is the number of clusters, and fi is the frequency of users that is categorized in cluster i for i = 1,2,...,k.The samples in the synthetic dataset are also categorized according to the original dataset's clustering centers, which summarize the users with all kinds of behaviors. Similarly, we have the distribution of the synthetic dataset

$$P_s = \{f_1', f_2', \ldots, f_k'\}.$$

$$KL(P_s||P_o) = \sum_{i=1}^{k} f_i \log \frac{f_i}{f_i'}.$$





(a) original data (b) $\varepsilon = \infty$ (non-privacy) (c) $\varepsilon = 5$ (d) $\varepsilon = 20$ (e) $\varepsilon = 80$