USENIX Security '26 Artifact Appendix: From Easy to Hard++: Promoting Differentially Private Image Synthesis Through Spatial-Frequency Curriculum

Chen Gong[†]
University of Virginia

Kecen Li[†]
University of Virginia

Zinan Lin
Microsoft Research

Tianhao Wang
University of Virginia

A Artifact Appendix

A.1 Abstract

This is the official implementation document of the Usenix Security paper, "From Easy to Hard++: Promoting Differentially Private Image Synthesis Through Time-Frequency Curriculum." In the `README.md` files of the repository[1], we provide a clear, point-by-point explanation that maps each table and figure to the corresponding code needed to generate its results. We provide a detailed description of how our repository matches each table and figure as follows.

A.2 Description & Requirements

Each table/figure is explicitly mapped in README.md, so reviewers can locate and run the exact implementation for any reported result. We isolate parameter configurations from implementations. Users can adjust them via "config" files.

A.2.1 Security, privacy, and ethical concerns

Our artifact is a research implementation for differentially private (DP) image synthesis and poses no security risks to evaluators' machines.

A.2.2 How to access

The DOI for the artifacts is on Zenodo.[2]

A.2.3 Hardware dependencies

- **GPU**: 4 NVIDIA GPUs with CUDA support (minimum: each GPU with 32GB VRAM).
- **CPU**: Multi-core CPU (8+ cores recommended) for data preprocessing and evaluation metrics.
- **Memory**: 32GB+ system RAM recommended for handling large batches and FID computation.
- **Storage**: 10GB free disk space for datasets, checkpoints, and generated samples.

A.2.4 Software dependencies

- **OS**: Linux
- **Python**: 3.9
- **Core packages** are listed in requirements.txt.
- **CUDA**: 11.8 or 12.1 (matching your PyTorch build)
- **Optional**: Docker (we provide a `Dockerfile` for containerized execution)

A.2.5 Benchmarks

- **Datasets:** all public, auto-download scripts are provided in `data_preparation.sh`
- **Evaluation metrics:** FID, classification accuracy.

A.3 Set-up

A.3.1 Installation

We provide a straightforward environment setup. The user only needs to run "bash install.sh" to install our environment in minutes, supporting availability and functionality assessments.

A.3.2 Basic Test

After activating the environment, execute: "python run.py setup.n_gpus_per_node=4 -m DP-FETA-Pro -dn mnist_28 -e 1.0 eval.mode=val pretrain.n_epochs_time=1 pretrain.n_epochs_freq=1 train.n_epochs=1 gen.data_num=100". Quantitative results will be logged in `exp/feta-pro/<exp_name>/stdout.txt`.

[1] https://github.com/2019ChenGong/Feta-Pro
[2] https://doi.org/10.5281/zenodo.17836341

A.4 Evaluation workflow

A.4.1 Major Claims

We tackle a central challenge in privacy-preserving ML: achieving a strong balance between utility and privacy without depending on potentially risky public images. DP-FETA-Pro introduces a novel framework that integrates spatial- and frequency-domain representations, offering new insights into the design of privacy-preserving generative models. DP-FETA-Pro achieves better FID and Acc than the state-of-the-art method. This is proven by the experiments.

A.4.2 Experiments

Environment: We provide a straightforward environment setup. The user only needs to run "bash install.sh" to install our environment in minutes, supporting availability and functionality assessments.

Data Preparation: This artifact provides a concrete data preparation process. All datasets used are publicly available and widely used in the community to avoid ethical concerns. The user should run "bash data_preparation.sh" to prepare all the studied datasets. Loading all datasets is time-consuming. For quick evaluation, we recommend loading only `MNIST` and `Fashion-MNIST`, using "bash data_preparation_quick.sh".

Main Contributions: We present how to match each table and figure in the main contributions of our paper as follows.

(1) *Reproducing Main Contributions (RQ1).* The strengths of our methods compared to baselines are reported in Table 4, Figure 3, and Figure 4. To train the DP-FETA-Pro synthesizer on the `MNIST` dataset with a privacy budget $\varepsilon = 1.0$ (as shown in Table 3), execute: "python run.py setup.n_gpus_per_node=4 –method DP-FETA-Pro –data_name mnist_28 -e 1.0 eval.mode=val".

To reproduce baseline results using the `DPImageBench` suite, modify the `-m` (method) and `-ed` (experiment description) flags. Supported baselines include `DP-MERF`, `DP-NTK`, `DP-Kernel`, `DPDM`, `DP-GAN`, `GS-WGAN`, and `DP-FETA`. Example for `DPDM`: "python run.py setup.n_gpus_per_node=4 setup.master_port=6662 -m DPDM -dn mnist_28 -e 1.0 -ed dpdm eval.mode=val".

Quantitative results of our experiments are logged in `exp/pdp-diffusion/<exp_name>/stdout.txt`. For Figure 3 (Visualizations) and Figure 4 (FID Curves), use: "python plot/visualization.py" for image quality visualization; and "python plot/fid_curve.py" for convergence analysis.

(2) *Ablation Studies (RQ2):* RQ2 investigates the benefits of frequency features and the auxiliary generator (Table 5 and Figure 5). Users can select frequency learning invariants via `pretrain.mode` as follows.

- `FETA-Pro_ft`: Prioritizes spatial then frequency (`freq_time`).
- `FETA-Pro_mix`: Simultaneous learning (`mix`).
- `FETA-Pro_f`: Frequency-only warm-up (`freq`).

Example command for "*mix*" mode: "python run.py pretrain.mode=mix -m DP-FETA-Pro -dn mnist_28 -e 1.0"

To compare with `No-Auxiliary` and `DM-Auxiliary` (Table 4), set `pretrain.mode` to `no_auxiliary` or `dm_auxiliary` respectively.

(3) *Privacy Budget Allocation (RQ3).* To reproduce the experiments of privacy allocation (Figures 6, 7, and Table 11): Adjust noise scales via `train.sigma_freq` (frequency domain) and `train.sigma_time` (spatial domain) as, "python run.py –method DP-FETA-Pro -e 1.0 train.sigma_freq=26.6 train.sigma_time=20". For Figure 7, users can adjust `-e` to run FETA-Pro with varying privacy budget.

Table 11 provides the RDP cost ratios. These can be computed directly using our analytical tool: "python cal_privacy.py –method DP-FETA-Pro -e 1.0 train.sigma_freq=26.6 train.sigma_time=20".

(4) *Extended Analysis and Discussion (results in Section 6.1 and Section 6.2).* To facilitate the experiments discussed in the extended analysis sections of the paper, we provide specialized scripts for non-private baselines and transfer learning scenarios. To establish a performance upper bound, the classification algorithm can be tested on sensitive images without DP protections is "python ./scripts/test_classifier.py –data_name mnist_28 –epsilon 10.0 -ed no-dp-mnist_28".

For experiments involving warm-up training with public images (as discussed in the Discussion section), set the pre-training mode to `time_freq`: "python run.py setup.n_gpus_per_node=3 pretrain.mode=time_freq –method PDP-Diffusion –data_name mnist_28 -e 10.0".

A.5 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2026/.