SoftwareX 33 (2026) 102526



Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx





Simet: Synthetic image metrics - a synthetic image evaluation framework

O. Agost[a,b,*], F. Aran[a,b], J. Rius[a], P. Fraile[a,b], I. Barri[b], J. Vilaplana[a], J. Mateo[a]

[a] Department of Computer Science, University of Lleida, Lleida, Spain
[b] GFT Technologies S.L.U., Madrid, Spain

ARTICLE INFO

Keywords:
Synthetic data
Evaluation
Framework
Metrics

ABSTRACT

Simet provides a modular framework designed for the rigorous evaluation of synthetic image datasets. The framework integrates data provisioning, preprocessing, feature extraction, and complementary metrics, including Fréchet Inception Distance (FID), generative Precision/Recall, and classifier two-sample area under the receiver operating characteristic curve (ROC-AUC), within a single GPU-accelerated pipeline. A restraint mechanism enables declarative pass or fail gating. YAML- and command-line (CLI)-driven orchestration, shared feature caches, and structured logs facilitate reproducible, continuous-integration (CI)-ready workflows. Extensible abstractions, including providers, transforms, feature extractors, and metrics, allow practitioners to add new data sources or tests with minimal code. Templates support downstream utility evaluations, such as training on synthetic data and testing on real data (TSTR). Simet is positioned relative to existing toolkits, and protocols are outlined to demonstrate scalable, multidimensional evaluation of synthetic image data.

Metadata

Code metadata

Nr.	Code metadata description	Metadata
C1	Current code version	v1.0.0post1
C2	Permanent link to code/repository used for this code version	https://github.com/dcg-udl-cat/simet
C4	Legal Code License	MIT
C5	Code versioning system used	Git
C6	Software code languages, tools, and services used	Python3, uv
C7	Compilation requirements, operating environments & dependencies	faiss, kagglehub, pyyaml, scikit-learn, torch, torchvision, typer
C8	If available Link to developer documentation/manual	https://simet.udl.cat
C9	Support email for questions	oriol.agost@udl.cat, oriol.agost@gft.com

1. Motivation and significance

Recent progress in generative modeling, from Generative Adversarial Networks (GANs) and Variational AutoEncoders (VAEs) to modern diffusion models, has enabled the creation of synthetic images that are nearly indistinguishable from real data to human observers. This realism, however, shifts the challenge from generation to evaluation: how do we rigorously assess when synthetic data can replace or complement real data? As realism improves, the challenge shifts from creating images to evaluating them. This progress raises a crucial question: *how can we determine when synthetic data are of sufficient quality to serve as a replacement or complement for real data?*

As synthetic realism improves, the need for rigorous evaluation methods has become crucial. For synthetic data to be trusted and valuable, its quality must be objectively quantified. Synthetic datasets

* Corresponding author at: Department of Computer Science, University of Lleida, Lleida, Spain.
Email addresses: oriol.agost@udl.cat (O. Agost), ferran.aran@udl.cat (F. Aran), josep.riustorrento@udl.cat (J. Rius), pablo.fraile@udl.cat (P. Fraile), ignasi.barri@gft.com (I. Barri), jordi.vilaplana@udl.cat (J. Vilaplana), jordi.mateo@udl.cat (J. Mateo).

https://doi.org/10.1016/j.softx.2026.102526
Received 14 November 2025; Received in revised form 15 January 2026; Accepted 16 January 2026
Available online 29 January 2026
2352-7110/© 2026 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

are increasingly used to train and evaluate vision models in practice (for example, in autonomous driving simulations) [1], making robust evaluation metrics and tools essential.

A variety of generative evaluation metrics have been proposed in the literature. *Fréchet Inception Distance* (FID) is the de facto standard for summarizing the distributional similarity between real and generated image sets [2]. However, FID is known to be sensitive to implementation details (e.g., image preprocessing and scaling); the CleanFID approach was introduced to standardize these steps and improve reproducibility across studies [3].

To better disentangle image quality aspects, *Precision* and *Recall* metrics for generative models were proposed [4], explicitly quantifying the fidelity-diversity trade-off (how reliably a generator produces realistic images versus how well it covers the variability of the real data). Classifier-based two-sample tests have also been explored: for instance, one can train a classifier to distinguish authentic vs. fake images and use its ROC-AUC as an indicator (with an AUC of 0.5 meaning synthetic data are indistinguishable from real) [5].

Beyond distributional measures, researchers increasingly advocate evaluating generative models via downstream task performance, such as *Train on Synthetic, Test on Real* (TSTR) experiments, to directly assess the utility of synthetic data in practical settings cite chung2025sok. Importantly, no single metric is sufficient to capture all facets of image quality [6]; each has strengths and blind spots, and consensus on a *best* metric remains elusive.

Despite the availability of various metrics, existing tools for synthetic image evaluation remain fragmented and often focus only on individual metrics. Researchers typically compute FID (and perhaps a few other scores) using standalone libraries or scripts, then separately analyze other aspects, which often leads to duplicated effort and potential inconsistencies. Several open-source libraries provide implementations of individual metrics (for example, PyTorch-based toolkits like TorchMetrics and CleanFID focus on standardizing FID/KID calculations [3], but these solutions do not offer an integrated end-to-end evaluation pipeline. For instance, `torch-fidelity` and `CleanFID` enable reliable computation of FID (and related scores) on given image sets, yet they handle only the metric computation in isolation. They assume the user will handle data loading, preprocessing, and combining results from multiple metrics, etc. This lack of integration and extensibility makes comprehensive evaluation tedious and error-prone.

SIMET was developed to address these gaps by providing a unified, extensible framework for evaluating synthetic image datasets. Compared to prior metric toolkits, SIMET emphasizes an end-to-end, dataset-centric pipeline with easy extensibility (e.g., drop-in new metrics or feature extractors). It automates data provisioning, feature extraction, and multi-metric scoring under a single roof, thereby streamlining the evaluation workflow. Table 1 contrasts the features of SIMET with two popular existing frameworks, `torch-fidelity` [7] and `CleanFID` [3], highlighting Simet's potential and capacity to fill the existing gaps.

Table 1
Feature comparison: SIMET vs. existing libraries.

Feature/Metric	SIMET (Ours)	torch-fidelity	CleanFID
FID (Inception)	Yes	Yes	Yes
Precision/Recall (gen. models)	Yes	Yes	No
ROC-AUC (C2ST)	Yes	No	No
TSTR/TRTS (downstream tasks)	Template-based	No	No
Pipeline (data loading + transforms)	Yes	Limited	Limited
Threshold-based gating	Yes	No	No
Extensibility (new metrics)	Yes	Partial	No

2. Software description

2.1. Software architecture

SIMET's design follows a modular pipeline architecture. Given a real dataset and a corresponding synthetic dataset, SIMET executes the following sequence of steps:

1. **Providers:** Interfaces for dataset access and iteration (e.g., directory readers for image folders, wrappers for subsampling or class-balanced reading, etc.). Providers abstract away the data source, presenting real and synthetic data uniformly to the rest of the pipeline.
2. **Transforms:** Pre-processing operations applied to images from each dataset. By default, SIMET applies an `InceptionTransform` (resizing images to 299×299 and normalizing pixel values) so that both real and synthetic images are processed into the input format expected by the default feature extractor. Users can configure custom transform chains if needed.
3. **Feature Extractors:** Any function that maps images into a feature space suitable for evaluation while preserving beneficial properties of the data (such as a deep network). SIMET uses the penultimate layer of a pretrained Inception-V3 network's (2048-D features) by default, which is standard for computing FID and related metrics. The framework supports swapping in alternative feature extractors (for example, a different convolutional backbone or a CLIP model) without changing the rest of the pipeline.
4. **Metrics:** A suite of metric computations is then applied to the extracted feature sets. Out of the box, SIMET includes implementations of several complementary metrics: FID (to assess global distributional closeness of synthetic to real), Precision and Recall (to measure fidelity and diversity of the synthetic set separately), and a two-sample classification test producing an ROC-AUC value (to gauge the discriminability of real vs. synthetic data). All metrics operate on the same extracted features for efficiency.
5. **Restraints:** Finally, SIMET can evaluate user-defined *threshold criteria* on the metrics. Each *restraint* corresponds to a condition (e.g., FID ≤ 30, Precision ≥ 0.70, ROC-AUC ≤ 0.60); The pipeline aggregates these and can produce an overall pass/fail judgment for the synthetic dataset. This allows quantitative gating, such as automatically flagging a synthetic dataset as unacceptable if it fails to achieve a required fidelity or has too high detectability.

This five-stage architecture is both extensible and configurable. Advanced users can plug in new providers (for novel data sources), custom feature extractors, or additional metrics (e.g., other image quality measures or task-specific metrics) by conforming to SIMET's interfaces. The pipeline will automatically incorporate these new components while preserving the overall flow. Both the abstractions and the supplied implementations can be seen in Fig. A.1 of Appendix A.

2.2. Software functionalities

SIMET provides a unified environment for the comprehensive evaluation of synthetic images. It implements multiple complementary evaluation metrics within one framework. In particular, SIMET can compute traditional distributional scores like FID, the generative Precision and Recall metrics (providing insight into fidelity vs. coverage), and a classifier-based two-sample test (reporting ROC-AUC), all in one run. Including these different metrics enables users to diagnose multiple aspects of generative performance simultaneously.

In addition, SIMET supports user-defined acceptance criteria for these metrics via the *restraint* mechanism, as described in Section 2.1. As stated, this enables automated pass/fail judgments, which are especially useful in production pipelines or continuous integration testing.

Beyond the metrics themselves, SIMET offers several practical features to facilitate use and integration. It provides a command-line interface (CLI) that orchestrates the entire evaluation pipeline via a

simple YAML configuration, as well as a Python API for programmatic use. Through the CLI or API, users can specify datasets, choose which metrics to run, set threshold criteria, and more.

The framework supports GPU acceleration for computationally intensive tasks, such as feature extraction and specific metric calculations, which are inherent to using Torch. It employs caching of intermediate results (like extracted features) to avoid redundant computation across runs. All metric outputs and run metadata (configs, seeds, metric details, dataset hashes) are recorded in structured logs, ensuring provenance and auditability across runs and supporting reproducible experiments by pinning the entire evaluation state.

Crucially, SIMET's modular design makes it straightforward to extend: researchers can add a new metric or test by implementing a small interface, and immediately benefit from the existing data loading, feature extraction, and logging infrastructure. This emphasis on extensibility means that new evaluation techniques can be integrated into SIMET without disrupting the overall system. In summary, SIMET's key functionalities include multi-metric evaluation, threshold-based dataset validation, user-friendly configuration and output, and a plugin-like extensibility model, which together greatly simplify the evaluation of synthetic image data.

3. Illustrative examples

In this section, we provide examples of all the ways to use the SIMET framework, showcasing both the Python API and the command line interface (CLI) capabilities. We provide three different CLI entry points, which can be used as shown in Listing 2.

- `simet simple`: This command allows users to quickly evaluate two datasets (real and synthetic) by providing their respective paths. It runs all available metrics by default, with no constraints. An example usage is shown in Listing 2.
- `simet pipeline`: This command allows users to define a custom evaluation pipeline using a YAML configuration file. This pipeline can include specific dataset providers, feature extractors, and constraints. An example configuration is shown in Listing 3.
- `simet simple-pipeline`: This command provides a simplified way to run evaluations using a YAML configuration file. It runs all metrics by default (see Listing 4), but users can also specify a subset of metrics to be computed (see Listing 5) It has no constraint capabilities.

For users who prefer to work directly with Python or use SIMET's full capabilities, Listing 1 demonstrates how to set up and run a custom

```python
# main.py
from pathlib import Path
from simet.dataset_loaders import DatasetLoader
from simet.feature_extractor import InceptionFeatureExtractor
from simet.pipeline import Pipeline
from simet.providers import LocalProviderWithoutClass
from simet.restraints import FIDRestraint, PrecisionRecallRestraint, RocAucRestraint
from simet.services import LoggingService, SeedingService
from simet.transforms import InceptionTransform

LoggingService.setup_logging()
SeedingService.set_global_seed(42)

loader = DatasetLoader(
    real_provider=LocalProviderWithoutClass(Path("data/real")),
    synth_provider=LocalProviderWithoutClass(Path("data/synth")),
    provider_transform=InceptionTransform(),
    feature_extractor=InceptionFeatureExtractor(),
)

pipeline = Pipeline(
    loader=loader,
    restraints=[
        FIDRestraint(upper_bound=40.0),
        PrecisionRecallRestraint(lower_bound=[0.70, 0.60]),
        RocAucRestraint(lower_bound=0.85),
    ],
)

ok = pipeline.run()
print("PASS" if ok else "FAIL")
```

Listing 1. A simple example of using the SIMET API to run a pipeline evaluating synthetic images against real images.

```
simet simple /path/to/real_images /path/to/synth_images  # Runs all metrics for two given paths
simet pipeline pipeline.yaml  # Runs a pipeline defined in a YAML config file
simet simple-pipeline simple_pipeline.yaml  # Runs a simplified pipeline defined in a YAML config file
```

Listing 2. Examples of using the SIMET command line interface to run evaluations.

```
# Pipeline.yaml
pipeline:
    loader:
        real_provider:
            type: LocalProviderWithoutClass
            path: my_real_imgs_path
        synth_provider:
            type: LocalProviderWithoutClass
            path: my_synth_imgs_path
        provider_transform:
            type: InceptionTransform
        feature_extractor:
            type: InceptionFeatureExtractor

    restraints:
        - type: FIDRestraint
          upper_bound: 30.0
          lower_bound: 0.0

        - type: PrecisionRecallRestraint
          upper_bound: [1.0, 1.0]   # [precision, recall]
          lower_bound: [0.7, 0.7]

        - type: RocAucRestraint
          upper_bound: 0.6
          lower_bound: 0.4
```

Listing 3. Example configuration of a config file running with restraints.

```
# simple_pipeline.yaml
pipeline:
    real_path: my_real_imgs_path
    synth_path: my_synth_imgs_path
```

Listing 4. Example configuration of a simplified pipeline config file running with all metrics.

```
# simple_pipeline.yaml
pipeline:
    real_path: my_real_imgs_path
    synth_path: my_synth_imgs_path

    metrics:
        - FID
        - PrecisionRecall
```

Listing 5. Example configuration of a simplified pipeline config file running with selected metrics.

evaluation pipeline using the SIMET API, using local dataset providers, an Inception feature extractor, and several constraints including FID, Precision-Recall, and ROC AUC.

4. Impact

SIMET opens up new possibilities for evaluating and improving generative models. A common challenge for researchers is determining whether improvements in FID correlate with increases in precision and recall (or other metrics) while simultaneously reducing detectability. It is essential to explore whether trade-offs exist between different aspects of synthetic data quality. Engineers need to meet predefined quality criteria to facilitate the deployment of reliable synthetic data generation systems. Constraints can be employed to automate this task. Finally, domain scientists can use SIMET's template-based approach for TSTR (Task-Specific Training Set Representation) and TRTS (Task-Relevant Test Set) evaluations to assess how effectively synthetic data supports specific downstream tasks, ensuring that synthetic datasets are fit for their intended purpose.

The YAML/CLI workflow and shared feature cache make these investigations routine, turning exploratory analyses into repeatable experiments and enforceable decisions. Using SIMET, it is straightforward to analyze correlations among metrics and identify cases where a model excels in one dimension of quality but falters in another. This capability can guide the development of more balanced generative models and inspire new research into combined metrics or evaluation protocols.

Additionally, since SIMET provides hooks for downstream task evaluation (such as TSTR/TRTS), researchers can directly examine how metric values relate to real-world utility (e.g., how much synthetic data degrades or improves task performance), an essential question in the field [8] . In short, SIMET helps bridge the gap between abstract metric scores and concrete task-level impacts, potentially leading to new insights and research directions in generative modeling.

The software also has a significant impact on the day-to-day practice of evaluating generative models. By offering a standardized, automated pipeline, SIMET improves the rigor and reproducibility of evaluation. Researchers no longer need to orchestrate disparate scripts to compute different metrics or worry about slight preprocessing differences across studies. SIMET ensures that all metrics share the exact data handling and that evaluations can be reproduced exactly via its configuration files and logging. This uniformity can enhance the comparability of results across papers.

Unlike libraries such as TORCH-FIDELITY, which center around metric-specific implementations and require invasive code changes to support new metrics, SIMET is designed for extensibility from the ground up. Its plugin-like architecture allows users to drop in custom metrics or feature extractors without modifying the core pipeline, making it especially well-suited to research environments where evaluation needs evolve rapidly.

Over time, as the community adopts tools like SIMET, evaluation criteria for synthetic data may become more consistent, addressing long-standing concerns about inconsistent evaluation protocols in generative modeling [6]. Moreover, SIMET 's extensibility means the framework can evolve with the field: as new metrics are proposed (for example, measures of image memorization or new fidelity metrics), they can be incorporated into the toolkit, allowing researchers to benefit from them immediately in a unified way. This flexibility ensures that the impact of SIMET will persist, as it can continuously integrate the best available evaluation practices.

From a practical standpoint, SIMET could change how generative models are developed and validated in both academic and industrial settings. Its threshold-based gating and automation-friendly design allow it to be inserted into model training pipelines or CI (continuous integration) systems. For instance, a development team can set up SIMET to automatically evaluate each new version of a generative model against a baseline real dataset; if the model's outputs do not meet the predetermined quality criteria (for example, if FID worsens or if any metric falls below the acceptable range), the system can flag this immediately. This kind of automated quality check can save considerable time and ensure that only models that improve or maintain synthetic data quality are promoted.

We anticipate that it will be helpful not only to generative modeling researchers seeking a comprehensive evaluation toolkit, but also to practitioners in fields such as graphics, simulation, or data augmentation who rely on synthetic images. In summary, by simplifying rigorous evaluation and integrating seamlessly into workflows, SIMET improves the pursuit of existing research questions (making evaluations more reliable and exhaustive) and lays the groundwork for exploring new questions about what *quality* means for synthetic data in practice.

5. Conclusions

We introduced SIMET, a modular, extensible, and scalable framework for evaluating synthetic image datasets against absolute references. Rather than serving as a collection of disjoint metric implementations, SIMET treats *synthetic dataset evaluation* as a first-class end-to-end problem. It unifies data access, preprocessing, feature extraction, multi-metric scoring, and decision-making under a coherent set of abstractions, enabling users to perform comprehensive evaluations with minimal effort.

By combining distributional metrics (FID, Precision/Recall) with discriminative tests (ROC-AUC), SIMET provides a rich, nuanced assessment of synthetic image quality. We demonstrated that using multiple metrics can reveal potential issues that might be overlooked by a single metric alone, and that SIMET's built-in threshold gating can automate the decision-making process of whether a synthetic dataset is of acceptable quality.

Moving forward, our goal is to keep expanding SIMET in step with the evolving needs of the community. In its current form, the framework is focused on image data and a core set of evaluation metrics; future versions will incorporate additional metrics (such as image *coverage* and *density* measures [9], improved perceptual quality scores, etc.) Beyond distributional and discriminative similarity, future work may integrate causality and semantics-aware metrics capable of quantifying whether generative models preserve underlying data-generating mechanisms and relational structure. Long-term, we aim to couple Simet with generative model training loops, creating a closed feedback system where evaluation outcomes directly guide adaptive optimization (e.g., metric-driven regeneration, iterative fine-tuning, and batch re-synthesis). We also plan to support other data modalities (e.g., synthetic audio or text data) using the same architectural principles and to add features such as automatic report generation (producing summary tables, plots, and example visuals) to simplify the analysis of results further.

Due to the design of SIMET's extensible framework, new tests, such as privacy or memorization checks to ensure generative models are not leaking sensitive training data can be integrated as additional restraint modules in the future. We envision SIMET becoming a living library that grows to accommodate emerging best practices in generative model evaluation. In conclusion, SIMET uniquely combines multi-metric evaluation, dataset-centric pipelines, threshold gating, and extensibility. By doing so, it substantially lowers the barrier to rigorous synthetic image assessment and contributes to more reliable and informative comparisons of generative models. We invite the community to use and contribute to SIMET, and we hope that it will serve as a valuable tool in advancing the science of synthetic data.

CRediT authorship contribution statement

O. Agost: Writing – original draft, Software, Methodology, Investigation, Conceptualization. **F. Aran:** Validation, Software. **J. Rius:** Writing – review & editing, Supervision, Project administration. **P. Fraile:** Validation, Software, Methodology. **I. Barri:** Writing – review & editing, Project administration, Conceptualization. **J. Vilaplana:**

Validation, Supervision, Methodology. **J. Mateo:** Writing – review & editing, Validation, Supervision, Software.

Code availability

The software described in this paper has been archived at Zenodo and is available at: https://zenodo.org/records/17609283 [10].

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

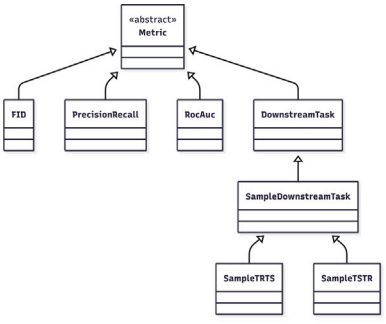
Acknowledgements

This work was supported by the Industrial Doctorates Plan of the Department of Research and Universities of the Government of Catalunya [grant number 2024 DI 00060].

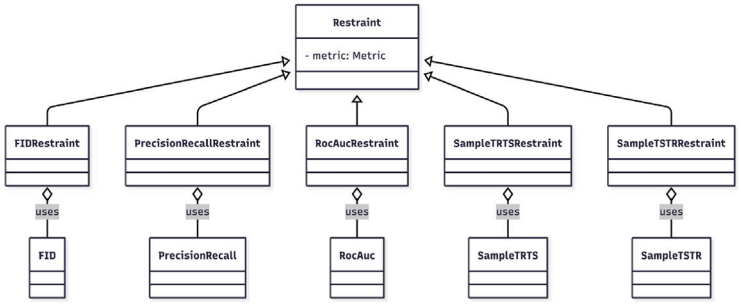
We thank GFT, in particular the AI Strategy initiative, for supporting this work by providing time, computing resources, and constructive feedback which made the project possible.

Appendix A. `Simet`'s architecture

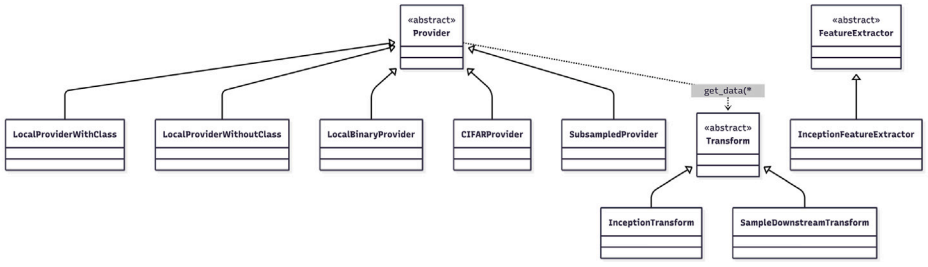
See Fig. A.1



(a) Metric architecture of the SIMET framework.



(b) Restraint architecture of the SIMET framework.



(c) Provider and feature extractor architecture of the SIMET framework.

Fig. A.1. Architecture of the SIMET framework.

References

[1] Gaidon A, Wang Q, Cabon Y, Vig E. Virtual Worlds as Proxy for Multi-Object Tracking Analysis. arXiv:1605.06457 May 2016.
[2] Heusel M, Ramsauer H, Unterthiner T, Nessler B, Hochreiter S. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In: Proceedings of the 31st international conference on neural information processing systems, NIPS'17. Red Hook, NY, USA: Curran Associates Inc.; 2017. p. 6629–40.
[3] Parmar G, Zhang R, Zhu J-Y. On Aliased Resizing and Surprising Subtleties in GAN Evaluation. arXiv:2104.11222 Jan 2022.
[4] Kynkäänniemi T, Karras T, Laine S, Lehtinen J, Aila T. Improved Precision and Recall Metric for Assessing Generative Models. arXiv:1904.06991 Oct 2019. https://doi.org/10.48550/arXiv.1904.06991
[5] Kim I, Ramdas A, Singh A, Wasserman L. Classification Accuracy as a Proxy for Two Sample Testing. arXiv:1602.02210 Feb 2020.
[6] Borji A. Pros and cons of GAN evaluation measures: new developments. Comput Vis Image Underst 2022;215:103329. https://doi.org/10.1016/j.cviu.2021.103329
[7] Torch fidelity – read the DOCS. https://torch-fidelity.readthedocs.io/en/latest/index.html.
[8] Chung Y, Zhang Y, Marrouche N, Hamm J. SoK: can synthetic images replace real data? A survey of utility and privacy of synthetic image generation. In: USENIX security symposium; 2025.
[9] Bińkowski M, Sutherland DJ, Arbel M, Gretton A. Demystifying MMD GANs. arXiv:1801.01401 Jan 2021.
[10] Agost O, Aran F, Rius J, Fraile P, Barri I, Vilaplana J, Mateo J. Simet: synthetic image metric evaluation; 2025. https://doi.org/10.5281/zenodo.17609282