WANGHLEY MARTINS
NICOLAS VASILESCU
LOGAN CHU

STEPDROP:

Stochastic Step Skipping in Tiny Diffusion Models

2025

Introduction

Diffusion models have recently emerged as one of the most powerful classes of generative models, capable of producing remarkably high-quality images through a process of iterative denoising. However, these models typically require large parameter counts and thousands of sampling steps to achieve their results, resulting in high computational costs and long inference times. In contrast, tiny diffusion models (those with fewer than 2–10 million parameters) offer a lightweight and computationally efficient alternative suitable for constrained environments such as mobile devices or embedded systems.

The goal of this project is to improve the speed and efficiency of tiny diffusion models without substantially compromising image quality. We propose to investigate a novel acceleration technique based on stochastic step skipping, in which the denoising process randomly omits certain timesteps according to a learnable or tunable probability distribution. By skipping redundant steps in a principled way, we aim to identify an optimal balance between image fidelity and computational speed, leading to faster inference and a deeper understanding of temporal redundancy in diffusion processes.

Objectives

Our primary objective is to compare the quality and performance of tiny diffusion models under different acceleration techniques. Specifically, we will:

1. Develop and train a baseline tiny diffusion model using standard DDPM (Denoising Diffusion Probabilistic Model) and DDIM (Denoising Diffusion Implicit Model) sampling techniques.
2. Implement a stochastic step-skip scheduler that probabilistically omits denoising steps during the sampling process.
3. Evaluate the performance of this stochastic scheduler against deterministic step-reduction methods using standard image quality metrics.

The central research question is:

> *Can stochastic step skipping reduce computation time and memory usage in lightweight diffusion models while maintaining comparable image quality to standard DDPM/DDIM approaches?*

Background and Motivation

In conventional diffusion models, each image is generated through a long, fixed sequence of denoising steps, where noise is gradually removed until a coherent image emerges. Although this sequential refinement ensures high fidelity, it is computationally expensive. Current acceleration techniques, such as DDIM or DPM-Solver, optimize or reduce the number of timesteps deterministically by using numerical solvers or fixed-step subsets. These approaches work well for large, high-capacity models but may not fully leverage the redundancy inherent in smaller networks.

Tiny diffusion models, due to their limited capacity, may inherently overrepresent certain timesteps or rely more heavily on specific temporal intervals. Introducing stochastic skipping could not only accelerate generation but also serve as a form of temporal regularization, analogous to dropout in neural networks. By preventing overreliance on specific time intervals, stochastic skipping might improve robustness and generalization, potentially revealing which timesteps are most essential for maintaining image quality. This concept has not been extensively explored in the current literature, particularly in the context of lightweight diffusion architectures.

Methodology

This project's methodology follows a structured four-week pipeline[1] combining literature review, implementation, experimentation, and analysis. In this first stage, the team reviews existing diffusion acceleration techniques and builds a baseline tiny diffusion acceleration techniques and builds a baseline tiny diffusion model (DDPM/DDIM) trained on CIFAR-10 or CelebA to establish a reference for image quality and speed.

Next, a stochastic step-skip scheduler is implemented to randomly omit denoising timesteps, either with fixed or learnable probabilities, and integrated into the model's sampling and training processes. Controlled experiments are then conducted to evaluate how different skip rates affect image fidelity, diversity, and computational latency, using Fréchet Inception Distance (FID), Inception Score (IS), and runtime measurements.

[1] See Appendix I for detailed information

Finally, results are analyzed through quantitative metrics and qualitative image comparisons, leading to a comprehensive report and presentation summarizing the trade-offs between speed and image quality achieved by stochastic skipping in tiny diffusion models.

Risks and Challenges

Tiny diffusion models have significantly fewer parameters than regular diffusion models, which means they can capture less complexity. This puts a greater emphasis on each step in the training process. Skipping steps may thus lead to a lower quality model.

Furthermore, the complexity of the skip scheduling may not be trivial and lead to degrading the process more when changes are applied.

We plan to utilize public, non-sensitive datasets to avoid data ethics/privacy concerns.

Evaluation Metrics

Performance will be assessed using three primary metrics that jointly capture image quality, diversity, and computational efficiency:

- Fréchet Inception Distance (FID)
    - FID measures how close the distribution of generated images is to that of real images, with lower scores indicating higher realism. For our project we aim achieving an FID within 10–20% of the baseline DDPM/DDIM model while significantly reducing the number of sampling steps. If stochastic step skipping can maintain a similar or only slightly higher FID while cutting inference time by 30–50%, this would demonstrate an effective balance between efficiency and quality.
- Inception Score (IS)
    - IS evaluates both the clarity and diversity of generated images, where higher scores are better. For success, our stochastic skip models should maintain an IS score comparable to or above 90% of the baseline. This would confirm that random step omission does not lead to mode collapse or a loss of image diversity,

ensuring that generated outputs remain visually distinct and semantically coherent.

- Latency and Step Count
    - We want to achieve a reduction in sampling time of at least 30% compared to the standard DDPM baseline, ideally without a substantial drop in FID or IS. Demonstrating faster generation with near-baseline quality would validate the practical benefit of stochastic step skipping for lightweight diffusion models.

Expected Outcomes

We expect the stochastic step-skip scheduler to demonstrate that:

- Selective skipping of diffusion timesteps can significantly reduce sampling time.
- Tiny diffusion models can tolerate moderate step reduction without severe degradation of image quality.
- Stochastic skipping may act as an implicit regularizer, improving robustness or generalization.

If successful, this project will provide both practical and theoretical insights into how temporal redundancy manifests in small-scale diffusion models. These findings could inform future designs of efficient generative models for real-time and edge-device applications.

References

Ho, J., Jain, A., & Abbeel, P. (2020). Denoising Diffusion Probabilistic Models. *Arxiv.org.* https://doi.org/10.48550/arXiv.2006.11239

lucidrains. (2023). *GitHub – lucidrains/denoising-diffusion-pytorch: Implementation of Denoising Diffusion Probabilistic Model in PyTorch.* GitHub. https://github.com/lucidrains/denoising-diffusion-pytorch

Stable Diffusion. (2022). *GitHub – CompVis/stable-diffusion: A latent text-to-image diffusion model.* GitHub. https://github.com/CompVis/stable-diffusion

tanelp. (2023). *GitHub - tanelp/tiny-diffusion: A minimal PyTorch implementation of probabilistic diffusion models for 2D datasets.* GitHub. https://github.com/tanelp/tiny-diffusion

TeaPearce. (2023). *GitHub – TeaPearce/Conditional_Diffusion_MNIST: Conditional diffusion model to generate MNIST. Minimal script.* GitHub. https://github.com/TeaPearce/Conditional_Diffusion_MNIST

Wang, Y., & Li, S. (2024). S2-DMs: Skip-Step Diffusion Models. *arXiv, 2401.01520.* https://arxiv.org/abs/2401.01520v2

Xue, S., Liu, Z., Chen, F., Zhang, S., Hu, T., & Xie, E. (2024). Accelerating Diffusion Sampling with Optimized Time Steps. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* IEEE. doi: 10.1109/CVPR52733.2024.00792

Zhang, L., & Ma, K. (2024). Accelerating Diffusion Models with One-to-Many Knowledge Distillation. ArXiv (Cornell University). https://doi.org/10.48550/arxiv.2410.04191

Zhu, Y., Liu, X., & Liu, Q. (2024). SlimFlow: Training Smaller One-Step Diffusion Models with Rectified Flow. *arXiv, 2407.12718.* https://arxiv.org/abs/2407.12718v2

Appendix I

Project Schedule

Step	Timeline (days)	Research Lead (Person A)	Implementation Lead (Person B)	Evaluation Lead (Person C)	Deliverables
0. Environment setup & repo	Day 1–2	Draft env spec, list required packages and references.	Create repo, CI/test scripts, conda/requirements, GPU config on Duke A100. Install libs (PyTorch, diffusers, pytorch-fid).	Prepare dataset download scripts and verify data accessibility for evaluation pipeline.	Repo scaffold, `requirements.txt`, working GPU env, dataset downloaded.
1. Baseline model & quick sanity runs	Day 3–7	Select baseline architecture (tiny U-Net ~2–10M), summarize hyperparams and β-schedule.	Implement model, training loop, checkpointing; run 1–2 quick epochs on CIFAR-10 to validate pipeline.	Integrate simple FID/IS script; generate initial sample grids from baseline.	Trained baseline checkpoint (toy), sample grid, working FID/IS script.
2. Sampling & deterministic baselines (DDPM/DDIM)	Day 8–10	Document deterministic sampling schedules to compare (full, reduced-evenly spaced).	Implement DDIM sampling and configurable step reduction; measure sampling wall-time and API.	Run baseline sampling, measure latency (#steps, wall time) and compute baseline FID/IS for short runs.	Baseline sampling scripts (DDPM, DDIM), baseline metrics table.

3. Design stochastic step-skip scheduler	Day 11–12	Propose scheduler types (fixed p, timestep-conditioned, learnable MLP) and math/pseudocode.	Create `/schedulers/skip_scheduler.py` API and integrate sampling-only skip mode.	Define evaluation protocol for skip experiments (seeded runs, sample size, logging).	Scheduler design doc, scheduler module (sampling-only) stub, eval protocol.
4. Sampling-time experiments (fast loop)	Day 13–15	Choose skip rates to test (e.g., 10%, 30%, 50%) and expected hypotheses.	Run sampling-only experiments with pretrained baseline using different skip rates; log outputs.	Compute FID/IS and latency for each skip rate; collate visual grids and metrics.	Short-report: sampling-only results (images + metrics) and runtime table.
5. Implement skip-aware training (if promising)	Day 16–20	Design training modifications (how to present skipped timesteps, loss adjustments, optional learnable scheduler training objective).	Modify training loop to sample with skip masks during training; add option for learnable scheduler module and checkpointing.	Monitor validation samples, track training logs (loss curves), and ensure metrics collection during training.	Trained skip-aware model(s) (checkpoints), training logs, implementation notes.
6. Main experiments & ablations	Day 21–24	Define experimental matrix (baseline, fixed-skip, learnable-skip; multiple skip rates). Prioritize runs for A100.	Schedule and run prioritized experiments on A100; automate job scripts; ensure reproducibility (seeds/configs).	Batch-evaluate all checkpoints: compute FID (5k if time allows), IS, and latency; produce	Full experiment dataset: checkpoints, CSV of metrics, raw images.

				CSV of results.	
7. Analysis & visualization	Day 25–26	Interpret results; write analysis of temporal importance and hypothesized mechanisms (regularization, redundancy).	Produce sample generation scripts for final visuals (high-quality grids) and timing scripts for latency measurement.	Create plots: FID vs steps, FID vs p_skip, latency reduction, sample comparisons; prepare figure captions.	Figure set (plots + sample grids), analysis bullets.
8. Report writing & presentation	Day 27–28	Draft Introduction, Related Work, Discussion, Conclusions, and References sections.	Write Methods and Implementation sections; attach config files and code pointers.	Compose Results section, include figures and metric tables; prepare slides and demo notebook.	Final report (≈2–8 pages), slides (10–12), demo notebook, reproducibility checklist.
9. Final polish & deliverables	Day 29–30	Final edit of report; prepare summary emails/README for reviewers.	Merge final code, tag release, upload checkpoints and short usage README.	Run final verification of all metrics and populate README with how to reproduce metrics.	Project deliverable bundle: report PDF, slides, code release, trained checkpoints, demo notebook.