DMin: Scalable Training Data Influence Estimation for Diffusion Models

Supplementary Material

7. Experimental Settings.

In this section, we report the detailed experimental settings and environments.

Implementation Details. We provide an open-source PyTorch implementation with multiprocessing support. We leverage Hugging Face, Accelerate, Transformers, Diffusers and PEFT in our implementation.

Experimental Environments. Our experiments are conducted on three types of servers: (1) Servers running Red Hat Enterprise Linux 7.8, equipped with Intel(R) Xeon(R) Platinum 8358 processors (2.60GHz) with 32 cores, 64 threads, 4 A100 80G GPUs, and 1TB of memory. (2) Servers running Red Hat Enterprise Linux 7.8, containing Intel(R) Xeon(R) Gold 6226R CPUs @ 2.90GHz with 16 cores, 32 threads, 2 A100 40G GPUs, and 754GB of memory. (3) A server running Ubuntu 20.04.6 LTS, featuring 2 H100 GPUs, dual Intel(R) Xeon(R) Gold 6438N processors (3.60GHz) with 32 cores, 64 threads, and 1.48TB of memory. To ensure a fair comparison, all experiments measuring time cost and memory consumption are conducted on server type (1), while other experiments are distributed across the different server types.

7.1. Models

This study evaluates the performance of the following models: (1) SD 1.4 with LoRA: This model integrates Stable Diffusion 1.4 (SD 1.4) with Low-Rank Adaptation (LoRA), a technique that fine-tunes large models efficiently by adapting specific layers to the target task while maintaining most of the original model's structure. (2) SD 3 Medium with LoRA: Utilizing the Stable Diffusion 3 Medium (SD 3 Medium) base model, this configuration applies LoRA for task-specific adaptation. The medium-sized architecture of SD 3 balances computational efficiency with high-quality generation performance. (3) SD 3 Medium: A standalone version of Stable Diffusion 3 Medium, serving as a baseline for comparison against the LoRA-enhanced models. This version operates without any additional fine-tuning, showcasing the model's capabilities in its default state. Additionally, we include the hyperparameter settings in Table 7.

7.2. Datasets

In this section, we introduce the datasets used in our experiments on conditional diffusion models and unconditional diffusion models.

Dataset Combination. For conditional diffusion models, we combine six datasets from Hugging Face: (1) magic-card-captions by clint-greene, (2) midjourney-detailed-prompts by MohamedRashad, (3) diffusiondb-2m-first-5k-canny by HighCWu, (4) lego-sets-latest by merve, (5) pokemon-blip-captions-en-ja by svjack, and (6) gesang-flowers by Albe-njupt. Additionally, we introduced noise to 5% of the data, selected randomly, and appended it to the dataset to enhance robustness. Finally, we split the data, allocating 80% (9,288 samples) for training and the remaining 20% for testing. For unconditional diffusion models, we use two classic datasets: (1) MNIST and (2) CIFAR-10.

Dataset Examples. Figure 4 showcases randomly selected examples from each dataset. For clarity, prompts are excluded from the visualizations. The original prompts can be accessed in the corresponding Hugging Face datasets.

Table 6. Average detection rate on different $\text{ef}_{\text{construction}}$, M and ef in HNSW implementation.

ef	Subset	M	$\text{ef}_{\text{construction}}$ 50	100	200	300	400	500
200	Flowers	4	0.8405	0.8349	0.8405	0.8405	0.8405	0.8405
		8	0.8410	0.8410	0.8415	0.8415	0.8415	0.8415
		16	0.8410	0.8415	0.8415	0.8415	0.8415	0.8415
		32	0.8410	0.8415	0.8415	0.8415	0.8415	0.8415
		48	0.8410	0.8415	0.8415	0.8415	0.8415	0.8415
	Lego Sets	4	0.2800	0.3035	0.3094	0.3094	0.3082	0.3082
		8	0.3082	0.3082	0.3094	0.3094	0.3094	0.3094
		16	0.3082	0.3082	0.3094	0.3082	0.3094	0.3094
		32	0.3082	0.3082	0.3094	0.3094	0.3094	0.3094
		48	0.3082	0.3082	0.3094	0.3094	0.3094	0.3094
	Magic Cards	4	0.9770	0.9772	0.9772	0.9772	0.9772	0.9772
		8	0.9772	0.9772	0.9772	0.9771	0.9771	0.9771
		16	0.9771	0.9771	0.9771	0.9771	0.9771	0.9771
		32	0.9771	0.9771	0.9771	0.9771	0.9771	0.9771
		48	0.9771	0.9771	0.9771	0.9771	0.9771	0.9771
1000	Flowers	4	0.8415	0.8415	0.8415	0.8415	0.8415	0.8415
		8	0.8415	0.8415	0.8415	0.8415	0.8415	0.8415
		16	0.8415	0.8415	0.8415	0.8415	0.8415	0.8415
		32	0.8415	0.8415	0.8415	0.8415	0.8415	0.8415
		48	0.8415	0.8415	0.8415	0.8415	0.8415	0.8415
	Lego Sets	4	0.2847	0.3035	0.3094	0.3094	0.3094	0.3094
		8	0.3094	0.3094	0.3094	0.3094	0.3094	0.3094
		16	0.3094	0.3094	0.3094	0.3094	0.3094	0.3094
		32	0.3094	0.3094	0.3094	0.3094	0.3094	0.3094
		48	0.3094	0.3094	0.3094	0.3094	0.3094	0.3094
	Magic Cards	4	0.9770	0.9771	0.9771	0.9771	0.9771	0.9771
		8	0.9771	0.9771	0.9771	0.9771	0.9771	0.9771
		16	0.9771	0.9771	0.9771	0.9771	0.9771	0.9771
		32	0.9771	0.9771	0.9771	0.9771	0.9771	0.9771
		48	0.9771	0.9771	0.9771	0.9771	0.9771	0.9771

8. Ablation Study

To better understand the impact of key parameters on the performance of the HNSW implementation, we conducted an ablation study by varying the graph-related parameters M and ef, as well as the construction parameter $\text{ef}_{\text{construction}}$. Table 6 summarizes the average detection rates across three subsets: Flowers, Lego Sets, and Magic Cards, under a

Figure 4. Examples of each dataset used in experiments.

Method	Learning Rate	Batch Size	# Epochs	Image Size	LoRA Rank	LoRA Alpha	LoRA Target Layers	Precision
SD 1.4 (LoRA)	0.001	64	150	512×512	4	8	[to_k, to_q, to_v, to_out.0]	float32
SD 3 Medium (LoRA)	0.001	64	150	512×512	4	8	[to_k, to_q, to_v, to_out.0]	float32
SD 3 Medium (Full)	0.0001	64	150	512×512	-	-	-	float32

Table 7. Hyperparameter settings for model training.

range of settings on SD 3 Medium with LoRA ($v = 2^{12}$).

The parameter M determines the maximum number of connections for each node in the graph. A larger M leads to denser graphs, which can improve accuracy at the cost of increased memory and computational overhead. The parameter $\text{ef}_{\text{construction}}$ controls the size of the dynamic list of candidates during graph construction, influencing how exhaustive the neighborhood exploration is during index creation. Lastly, the query-time parameter ef defines the size of the candidate list used during the search operation, directly affecting the trade-off between accuracy and efficiency.

Across the three datasets, the Magic Cards consistently exhibited high detection rates, exceeding 97.7% in all configurations, indicating that it is less sensitive to parameter tuning. In contrast, the Lego Sets showed significant variability. For $\text{ef} = 200$, the detection rate improved notably with higher values of M (e.g., from 28% at $M = 4$ to 30.82% at $M = 8$ in $\text{ef} = 200$ and $\text{ef}_{\text{construction}} = 50$), but beyond $\text{ef}_{\text{construction}} = 100$, further increases in $\text{ef}_{\text{construction}}$ provided diminishing returns. This suggests that while denser graphs and more exhaustive index construction improve accuracy for complex datasets, the benefits plateau at a certain point. For the Flowers, the detection rates remained stable at approximately 84.1% across all parameter settings, indicating that this dataset is robust to variations in M and ef.

9. Additional Analysis

Training order and data augmentation. Our influence score is defined with respect to the final trained parameters θ, rather than intermediate checkpoints. While training order affects the optimization trajectory, its effect is absorbed into the final parameter state used for influence estimation, especially under large-scale training with extensive shuffling. Data augmentation is treated as part of the training

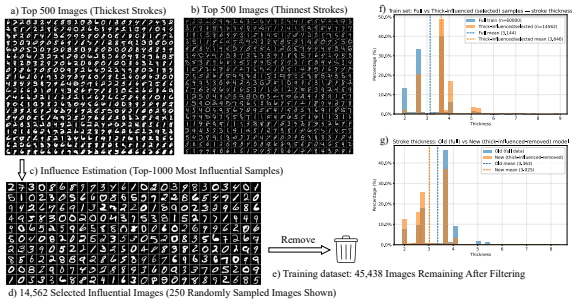

Figure 5. Removing highly influential MNIST training samples systematically shifts generated digits from thick to thinner strokes.

distribution; in practice, we report influence at the image level instead of treating each augmented view as a separate training sample.

Timestep subsampling and gradient correlation. Adjacent diffusion timesteps are often highly correlated. `DMin` accounts for this by subsampling timesteps for influence estimation instead of computing gradients for all T timesteps, as described in the main method. This design reduces computation and storage while preserving performance.

Effect of removing influential samples. We further study whether removing highly influential samples changes model behavior on MNIST. We first train an unconditional DDPM on the full 60,000-image training set and generate 5,000 samples. Using a stroke-thickness metric, we select 500 generated images with the thickest strokes, run top-1000 influence estimation for each image, and deduplicate the retrieved training samples, yielding 14,562 influential samples. After removing these samples and retraining on the remaining 45,438 images, generated digits shift toward thinner strokes, indicating that removing highly influential samples leads to systematic and interpretable changes in generation behavior (Figure 5).

9.1. Baselines

We compare the proposed `DMin` with seven baselines:

- **Random Selection:** Serves as a simple yet essential baseline where data points are selected randomly. This approach tests the performance against non-informed selection methods and ensures fairness in evaluation.
- **SSIM:** A widely-used metric for assessing the similarity between two images or signals. This baseline tests the performance of similarity measures rooted in visual or structural fidelity.
- **CLIP Similarity:** Exploits the feature embeddings generated by the CLIP, comparing their cosine similarity. It assesses how well general-purpose visual-language models can capture meaningful data relationships.
- **LiSSA:** Measures the influence of training points on the model's predictions by linearizing the loss function. This baseline provides a data-centric perspective on sample selection based on their impact on model training.

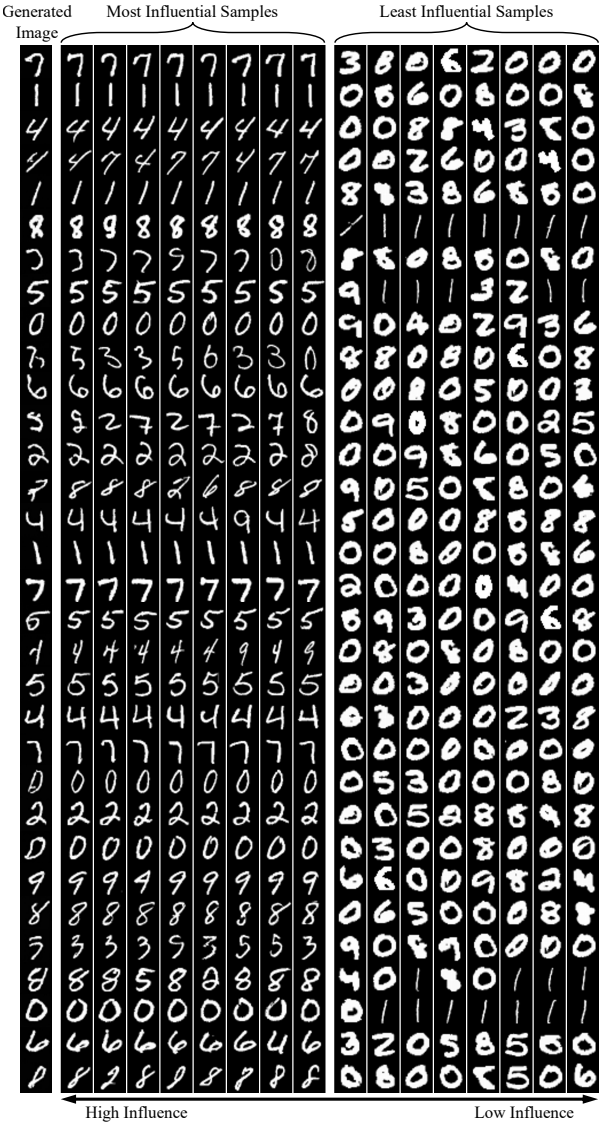

Figure 6. Additional visualizations for unconditional diffusion models on the MNIST dataset.

- **DataInf:** Employs data influence techniques to prioritize training samples that most strongly influence specific predictions. It represents methods that utilize influence diagnostics in data selection.
- **D-TRAK:** Focuses on tracking data's training impact using gradient information. This baseline evaluates approaches that harness gradient dynamics for data importance measurement.
- **Journey-TRAK:** Similar to D-TRAK but extends it to capture cumulative training effects over extended iterations. It benchmarks the ability of methods to consider long-term training trajectories in sample importance.

10. Supplemental Visualization for Conditional Diffusion Models

We provide additional visualizations for unconditional models on the MNIST dataset in Figure 6 and for conditional models in Figure 7. Examples for other methods are omitted as they are nearly identical.

Figure 7. Examples of the top-25 most influential training data samples for the generated image (first column) on SD 3 Medium with LoRA, shown from high to low influence from left to right.