Accelerating Diffusion Model Training under Minimal Budgets: A Condensation-Based Perspective

Supplementary Material

A. Positioning D^{2}C within Dataset Condensation Paradigms

While some works equate dataset condensation with gradient-based pixel-level optimization of synthetic images, a broader line of literature defines it as constructing compact training sets that retain the learning efficacy of the original data [25], which also includes image-level schemes such as OD3[32] and RDED[30]. In this broader paradigm, the key objective is not how the condensed data are obtained, but whether the resulting small dataset can support training models that closely match the performance of those trained on the full dataset. D^{2}C follows this latter view. It condenses the dataset by selecting a highly informative subset guided by diffusion difficulty and then attaching additional semantic and visual information that enriches each sample without altering its raw pixels. This design is analogous in spirit to OD3 and RDED, which also operate at the level of image selection rather than direct pixel optimization. Consequently, D^{2}C naturally fits within the dataset condensation family, while being specifically tailored to generative diffusion models and addressing a gap that is not covered by existing pixel-level condensation methods.

B. Additional Descriptions of Diffusion Models

This section reviews the fundamentals of the Denoising Diffusion Probabilistic Model (DDPM) [3]. The DDPM framework consists of a fixed forward process that incrementally perturbs the input data with noise, and a learned reverse process trained to iteratively denoise the data, thereby learning the target distribution. Specific architectural details of our implementation are summarized in Appendix B.2.

B.1. Denoising Diffusion Probabilistic Model

The DDPM framework models data generation via a discrete-time Markov chain that progressively adds Gaussian noise to a data sample $x_0 \sim p(x)$. The forward process is defined as:

$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t \mathbf{I}), \tag{13}$$

where $\beta_t \in (0,1)$ are predefined variance schedule parameters controlling the noise level at each time step $t \in [1, 2, ..., T]$, and $\mathbf{I}$ is the identity matrix.

For simplicity, we define $\alpha_t = 1 - \beta_t$, and denote the cumulative product $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$. The reverse process, which is learned by the model θ, can be defined as:

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}\left(x_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right), \Sigma_\theta(x_t, t)\right), \tag{14}$$

where $\epsilon_\theta(x_t, t)$ denotes the predicted noise from a neural network. The covariance $\Sigma_\theta(x_t, t)$ is typically set to $\sigma_t^2\mathbf{I}$, where σ_t^2 can be either fixed ($\sigma_t^2 = \beta_t$) or learned through interpolation $\sigma_t^2 = (1-\bar{\alpha}_{t-1})/(1-\bar{\alpha}_t)\beta$.

A simplified training objective minimizes the prediction error between true and estimated noise:

$$\mathcal{L}_{\text{simple}} = \mathbb{E}_{x_0,\epsilon,t}\left[\|\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t\right)\|^2\right]. \tag{15}$$

In addition to the simple objective, improved variants include learning the reverse variance $\Sigma_\theta(x_t, t)$ jointly with the mean, which leads to a variational bound loss of the form:

$$\mathcal{L}_{\text{vlb}} = \exp\left(v \log \beta_t + (1-v)\log \tilde{\beta}_t\right). \tag{16}$$

Here, v is an element-wise weight across model output dimensions. When T is sufficiently large and the noise schedule is carefully chosen, the terminal distribution $p(x_T)$ approximates an isotropic Gaussian. Sampling is then performed by iteratively applying the learned reverse process to recover the data sample from pure noise.

B.2. Diffusion Transformer Architecture

Our model implementation closely follows the design of DiT [9] and SiT [11], which extend the vision transformer (ViT) architecture [46] to generative modeling. An input image is first split into patches, reshaped into a 1D sequence of length N, and then processed through transformer layers. To reduce spatial resolution and computational cost, we follow prior work [9, 11] and encode the image into a latent tensor $z = E(x)$ using a pretrained encoder E from the stable diffusion VAE.

In contrast to the standard ViT, our transformer blocks include time-aware adaptive normalization layers known as adaLN-zero. These layers scale and shift the hidden state in each attention block according to the diffusion timestep and conditioning signals. During training, we also add an auxiliary multilayer perceptron (MLP) head that maps the hidden state to a semantic target representation space, such as DINOv2 [40] or CLIP features [47]. This head is used only for training-time supervision in our alignment loss and does not affect sampling or inference.

C. Hyperparameters and Implementation Details

Select Phase Settings. In the *Select* phase, we adopt a pretrained DiT-XL/2 model [9] as the scoring network and use the diffusion loss (*w.r.t.*, mean squared error) as the scoring metric. To construct subsets of different sizes, we apply interval sampling with $k = 96$ for the 10K subset, $k = 16$ for the 50K subset, and $k = 10$ for the 100K subset. Each subset is constructed in a class-wise manner, selecting 10, 50, and 100 samples per class respectively.

Attach Phase Settings. In the *Attach* phase, we implement dual conditional embeddings. For textual conditioning, we use a T5 encoder [39] with captions truncated to 16 tokens, producing embeddings of dimension 2048. For visual conditioning, we adopt DINOv2-L [40] as the visual encoder. The number of visual tokens h is set to 256, and each token has a feature dimension of 768.

Training Settings. In the *Training* phase, we use the Adam optimizer with a fixed learning rate of 1e-4 and $(\beta_1, \beta_2) = (0.9, 0.999)$, without applying weight decay. We employ mixed-precision (fp16) training with gradient clipping. Latent representations are pre-computed using the stable diffusion VAE, and decoded via its native decoder. All experiments are conducted on either 8 NVIDIA A800 80GB GPUs or 8 NVIDIA RTX 4090 24GB GPUs. We use a batch size of 256 with a 256×256 resolution in Fig. 1, and a 512×512 resolution in Table 2. All other experiments use a batch size of 128 and a default image resolution of 256×256.

D. Evaluation Details

We adopt several widely used metrics to evaluate generation quality and diversity:

- **gFID** [43] computes the Fréchet distance between the feature distributions of real and generated images. Features are extracted using the Inception-v3 network [48].
- **sFID** [49] extends FID by leveraging intermediate spatial features from the Inception-v3 model to better capture spatial structure and style in generated images.
- **IS** [44] evaluates both the quality and diversity of generated samples by computing the KL-divergence between the conditional label distribution and the marginal distribution over predicted classes, using softmax-normalized logits.
- **Precision and Recall** [50] respectively measure sample realism and diversity, quantifying how well generated samples cover the data manifold and vice versa.

E. Baseline Setting

We evaluate our method against two categories of baselines:

Diffusion models trained on selected or condensed subsets. These include SiT and DiT backbones trained from scratch on 10K, 50K, and 100K subsets obtained via the following strategies:

- **Random Sampling.** A naive baseline that randomly selects a fixed number of real samples without any guidance.
- **Herding** [36]. A geometry-based method that selects samples to approximate the global feature mean, ensuring representative coverage.
- **K-Center** [35]. A diversity-focused algorithm that iteratively selects samples maximizing the minimum distance from the selected set, promoting broad coverage of the feature space.
- **SRe^{2}L** [27]. A dataset condensation method that synthesizes class-conditional data through a multi-stage pipeline. Originally proposed for classification tasks, we adapt it to the diffusion setting by applying class-wise condensation to real images and training a diffusion model on the resulting synthetic subset. Visualizations of the synthesized samples and corresponding training results are provided in Appendix K.

Diffusion models trained on the full dataset. These baselines are trained with access to the entire training set, without data reduction:

- **SiT** [11]. A transformer-based diffusion model that reformulates denoising as continuous stochastic interpolation, enabling faster training and improved efficiency under full-data settings.
- **REPA** [10]. A model-side regularization method that aligns intermediate features of diffusion transformers with patch-wise representations from strong pretrained visual encoders (e.g., DINOv2-L [40], MAE [51], MoCov3 [52]) using a contrastive loss. It retains the full dataset and improves convergence and generation quality via early-layer representation guidance.

F. Framework Design and Implementation

We introduce D^{2}C, a framework for constructing compact yet effective training subsets for diffusion models under stringent data budgets. Our approach is motivated by two complementary intuitions: (1) that the contribution of training samples is non-uniform, as some are more informative than others; and (2) that generative training benefits from semantically enriched conditioning. These insights directly inform the two core stages of our framework. First, a *Select* stage ranks training examples by a difficulty score computed via a pretrained class-conditional diffusion model. Second, an *Attach* stage enriches the selected data by injecting textual and visual priors. The complete pipeline is summarized in Algorithm 1.

Algorithm 1 D^{2}C: Diffusion Dataset Condensation

Require: Full dataset $\mathcal{D} = \{(x_i, c_i)\}_{i=1}^{N}$, interval k, text encoder f_{text}, visual encoder f_{vis}
// Each x_i is an image, and $c_i \in \{1, \ldots, C\}$ is the class label.

// Phase 1: Select
Compute difficulty score s_{diff} for all $(x_i, c_i) \in \mathcal{D}$
For each class c, sort $\mathcal{D}_c = \{x_i \mid c_i = c\}$ by s_{diff} ascending
Select every k-th sample (Interval Sampling) in sorted $\mathcal{D}_c$ to form $\mathcal{D}_{\text{select}}$
// Phase 2: Attach
for each $(x, c) \in \mathcal{D}_{\text{select}}$ **do**
  Generate class prompt $P(c)$ (e.g., "a photo of a `label`")
  Extract text embedding: $(t_c, t_{\text{mask}}) \leftarrow f_{\text{text}}(P(c))$
  Extract visual feature: $y_{\text{vis}} \leftarrow f_{\text{vis}}(x)$
  Store tuple $(x, c, t_c, t_{\text{mask}}, y_{\text{vis}})$ into $\widetilde{\mathcal{D}}$
end for
Return enriched dataset $\widetilde{\mathcal{D}}$ for diffusion model training

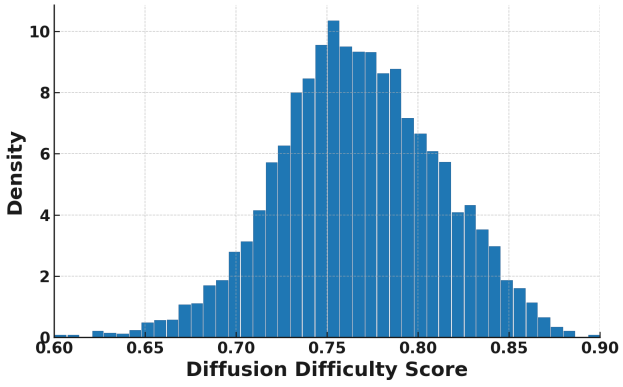

Figure 7. Distribution of diffusion difficulty score computed on LAION text–image pairs with a pre-trained SDXL model. This distribution resembles that of C2I, which supports interval sampling for selecting informative training pairs under T2I.

G. Exploration on Text-to-Image Generation

We further examine the applicability of the ***D^{2}C*** framework to text-to-image generation [53–55]. The *Select* phase requires only a minimal change: replace the class condition in Eq. 7 with a text condition, i.e., $s_{\text{diff}}^{text}(x) = -p_\theta(x \mid \text{text})$. Using SDXL to score LAION text–image pairs, we observe a difficulty distribution similar to the class-conditional case (Fig. 7; see also Fig. 3 and Fig. 8 (right)). Low-score samples tend to exhibit simple structures, high-score samples often contain complex or cluttered contexts, and the majority of samples fall in the middle range. Interval sampling remains effective for identifying informative pairs. The *Attach* phase is also easy to transfer: semantic and visual representations serve as soft supervisory signals for the selected subset.

As such, while our main experiments focus on class-to-image tasks for controlled benchmarking like SiT [11], the framework is generalizable and well suited to text-to-image generation. We expect it to deliver practical gains in data efficiency and training speed in this setting, offering a promising direction for future work.

H. More Discussions about Select

H.1. Detailed Algorithm for Computing Diffusion Difficulty Score

The diffusion difficulty score, used to rank samples in the *Select* phase, is defined as the mean denoising loss over uniformly sampled timesteps, computed using a frozen pre-trained diffusion model (see Algorithm 2).

Algorithm 2 Compute Diffusion Difficulty Score

Require: Image dataset $\mathcal{D} = \{(x_i, c_i)\}_{i=1}^{N}$; pretrained VAE encoder E_ϕ; pretrained diffusion model ϵ_θ; timestep set $\mathcal{T}$; batch size n
// Each x_i is an image; $c_i \in \{1, \ldots, C\}$ is the class label. Timesteps in $\mathcal{T}$ are sampled uniformly. Models are frozen during scoring.

Initialize empty map $\mathcal{S} \leftarrow \{\}$
for mini-batch $\{(x_i, c_i)\}_{i=1}^{n} \subset \mathcal{D}$ **do**
  Encode to latent (if applicable): $z_i \leftarrow E_\phi(x_i)$
  Initialize per-sample accumulator $\ell_i \leftarrow 0$
  for $t \in \mathcal{T}$ **do**
    Sample $\epsilon \sim \mathcal{N}(0, I)$
    Perturb latent: $z_t \leftarrow \alpha_t\, z_i + \sigma_t\, \epsilon$
    Compute loss: $\ell_i \leftarrow \ell_i + \|\epsilon - \epsilon_\theta(z_t, t, c_i)\|_2^2$
  end for
  $s_i \leftarrow \ell_i / |\mathcal{T}|$ *// Mean denoising loss across timesteps*
  $\mathcal{S}[x_i] \leftarrow s_i$
end for
Return $\mathcal{S}$ *// Image-to-score mapping for difficulty-aware selection*

H.2. Practical Insights on Interval Sampling

While Section 4.3 has covered a detailed ablation study on the choice of interval k in *Select* phase, we provide additional insights into how diffusion difficulty score relates to distributional coverage.

The right panel in Fig. 8 presents the gFID-10K scores of subsets sampled from different portions of the difficulty-ranked dataset. We partition the training set into consecutive 10K segments ordered by the diffusion difficulty score

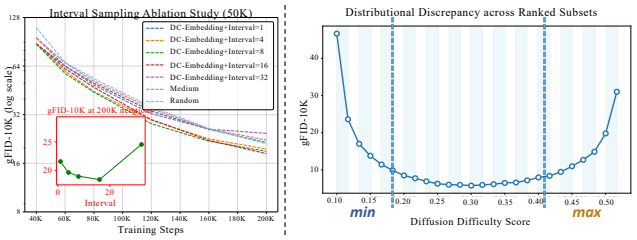

Figure 8. **Left:** gFID-10K across training steps under different interval values k for a 50K data budget. Moderate intervals (e.g., $k = 16$) achieve superior performance by balancing learnability and diversity. **Right:** Distributional discrepancy (gFID-10K) between ranked training subsets and the validation set. Both extremely low and high diffusion difficulty score lead to higher FID, while mid-range segments show better alignment.

(e.g., the first 10K samples with lowest scores as "Min", followed by 10–20K, 20–30K, and so on), and measure each segment's discrepancy from the full validation distribution using gFID. Interestingly, we observe a clear U-shaped curve: subsets consisting of extremely low or high difficulty samples exhibit significantly worse distributional alignment, while those centered around moderate difficulty levels show substantially lower FID scores. This result aligns well with our hypothesis that very easy samples (e.g., simple textures, clean backgrounds) and extremely hard samples (e.g., ambiguous, noisy structures) both fail to reflect the global data distribution.

These observations provide an empirical justification for our interval sampling strategy. Specifically, under a 50K dataset budget with $k = 16$, each class contributes samples selected at regular intervals from its difficulty-sorted list. Given that each class typically contains around 1,200 images, this strategy naturally samples from approximately the first 800 positions in the ranked list. As a result, the selected data span both the easy and moderately difficult regions, while avoiding the extremes at both ends. This balanced coverage across the difficulty spectrum promotes better generalization and faster convergence, as evidenced by the results in Fig. 8 (Left) and discussed in Section 4.3. In this way, our strategy yields a compact yet effective dataset that enables the model to converge rapidly while maintaining strong generation quality.

Ablation on interval sampling. As shown in Fig. 9, the "Medium" variant corresponds to selecting samples from the center of the difficulty-ranked list rather than applying interval sampling from low to high diffusion difficulty scores. Concretely, after sorting each class by diffusion difficulty, we start from the median position and expand symmetrically toward both sides until the data budget is reached. This strategy focuses on medium-difficulty examples and largely omits easier instances, while still including a portion of harder ones near the tails. As a result, the selected subset provides less comprehensive coverage of the

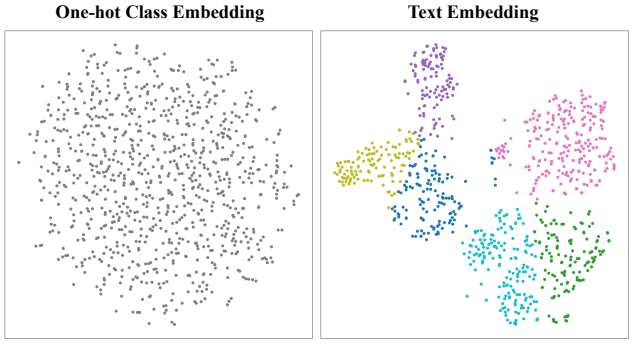

Figure 9. T-SNE visualization of class embeddings. Each point represents a class in the dataset. **Left:** One-hot class embeddings show no semantic structure. **Right:** Text embeddings naturally cluster semantically related classes. Samples from semantically related classes, such as different dog breeds, tend to form distinct clusters in feature space. Leveraging this semantic prior is highly effective for accelerating diffusion model training.

underlying data distribution, leading to slower convergence and degraded final performance compared to our proposed interval sampling scheme.

I. More Discussions about Attach

I.1. Dual Conditional Embedding

Most diffusion models condition on class identifiers represented as integer IDs or one-hot vectors, which are mapped to class embeddings trained from scratch. This ignores semantic relationships between categories, resulting in unstructured embeddings as shown in Fig. 9 (Left).In contrast, text embeddings derived from class-specific prompts (e.g., "a photo of a dog") via a pre-trained language encoder naturally encode semantic priors and cluster related classes (Fig. 9, Right). We propose a dual conditional embedding that fuses the text embedding with a learnable class embedding (i.e., a traditional class token trained from scratch), as defined in Eq. 8–9. This hybrid strategy combines semantic structure with symbolic distinctiveness, and leads to significantly improved generation quality. As shown in Fig. 6 (Right), using both branches achieves lower FID than using either one alone.

I.2. Visual Information Injection

Recent studies [10, 19] have shown that relying solely on diffusion models to learn meaningful representations from scratch often results in suboptimal semantic features. In contrast, injecting high-quality visual priors, especially those derived from strong self-supervised encoders like DINOv2 [40], can significantly improve both training efficiency and generation quality. In our case, we incorporate a frozen visual encoder to provide external patch-level visual features during training. These external features serve

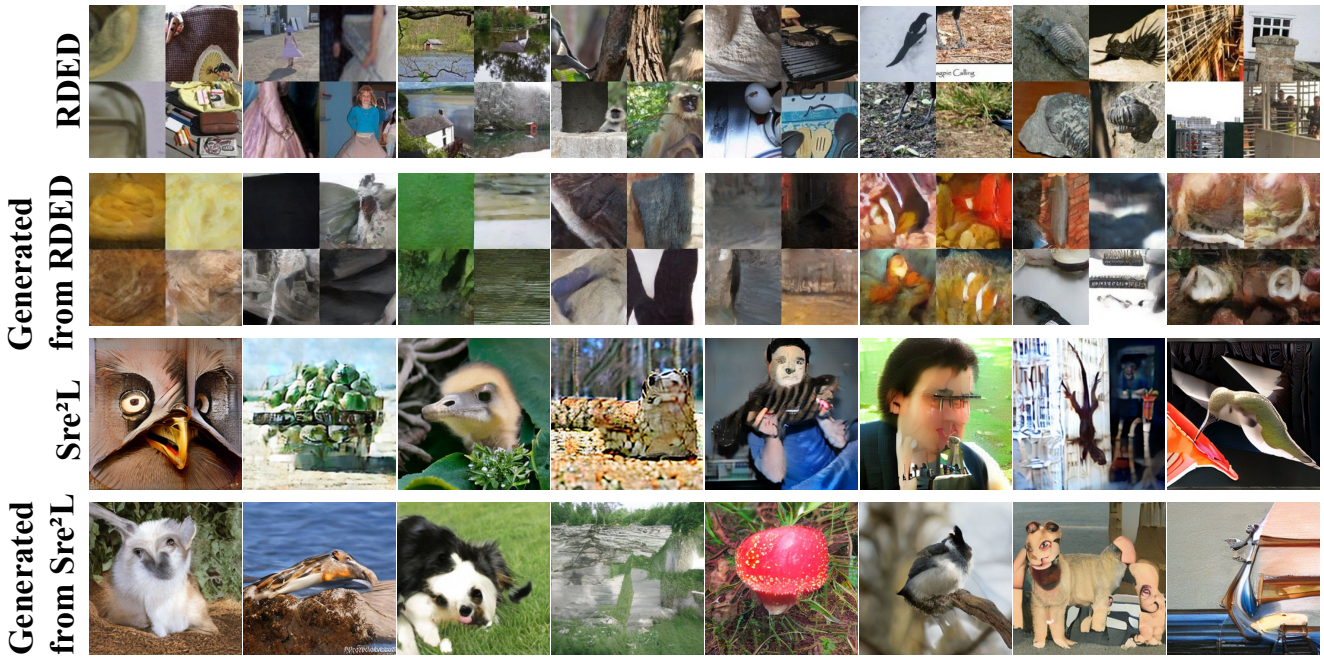

Figure 10. **Top:** Images synthesized directly by SRe^{2}L and RDED, two popular dataset condensation methods originally designed for discriminative tasks. **Bottom:** Images generated by diffusion model trained on the two synthesized datasets.

Table 7. Ablation of the visual encoder.

Vision Encoder	FID↓
N/A (baseline)	37.07
MAE-L	9.23
MoCov3-L	8.78
CLIP-L	8.59
DINOv2-L	**7.62**

as semantically rich anchors, particularly beneficial at early layers, allowing the model to focus on generation-specific details in later stages. Empirically, visual supervision improves feature alignment and accelerates convergence under limited data, as shown in Tables 1, 2, 5, and 7. All tested encoders outperform the no-encoder baseline, indicating that our method is robust to the choice of visual encoder.

J. Experiments on CIFAR

As shown in Table 8, we further evaluate D^2C on CIFAR-10 by selecting 100 images per class to form a 1K data budget (2% compression rate) and training the diffusion model for 100k steps. Under this highly constrained setting, D^2C significantly improves gFID from 9.72 with random sampling to 3.95, demonstrating that our selection and attachment strategy remains effective beyond ImageNet and transfers well across datasets.

Table 8. Comparison of random subset selection and D^2C on CIFAR-10 (reported in gFID-50K).

Method	gFID↓
Random	9.72
D^2C (Ours)	**3.95**

K. Visualization of SRe^{2}L and RDED in Generative Tasks

As shown in Fig. 10, dataset condensation methods that excel in classification, such as RDED and SRe^{2}L, transfer poorly to diffusion-based generation. Their objectives focus on preserving class-discriminative cues, for example segmentation-guided selection in RDED and gradient-based image optimization in SRe^{2}L, rather than modeling realistic global structure and natural image statistics. As a result, diffusion models trained on these synthesized datasets fail to capture the underlying pixel-level data distribution and produce severely degraded samples. In contrast, D^{2}C provides the first dataset condensation framework tailored to diffusion generative modeling and effectively closes this gap.

L. ImageNet 512×512 Experiment

As shown in Table 2, D^2C consistently outperforms random sampling under a strict 10K (0.8%) data budget across both DiT-L/2 and SiT-L/2 backbones. Visual samples in Fig. 11

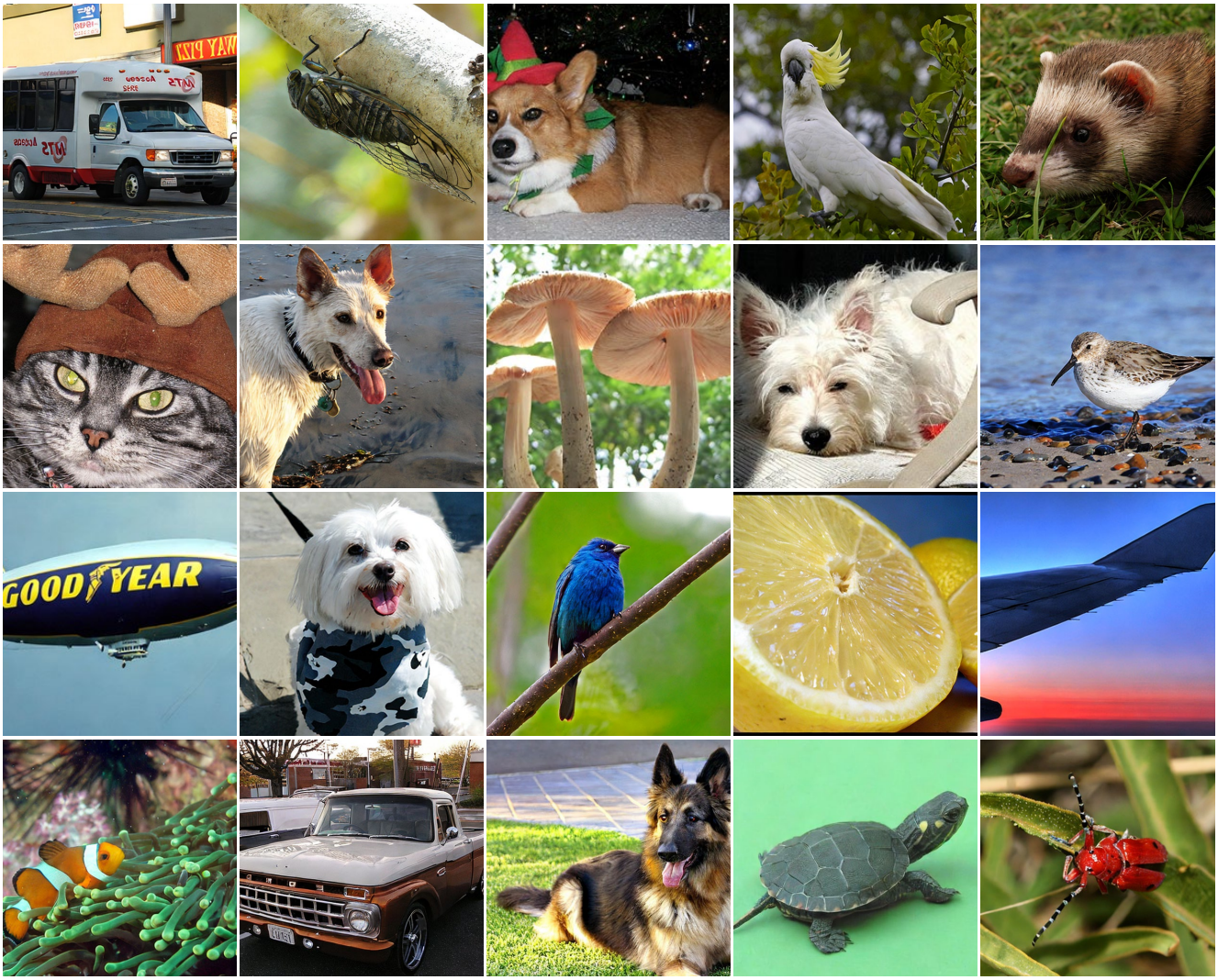

Figure 11. Generated samples on ImageNet 512×512 from SiT-L/2 trained with D^2C using a 10K dataset (CFG=1.5).

further confirm the high fidelity and diversity of generations at 512×512 resolution, demonstrating that D^2C generalizes effectively to high-resolution settings.

M. Visualization

Figure 12. Generated samples of SiT-L/2 trained with D^2C using a 50K dataset (CFG=1.5). Class label = ”macaw”(88)

Figure 13. Generated samples of SiT-L/2 trained with D^2C using a 50K dataset (CFG=1.5). Class label = ”arctic wolf”(270)

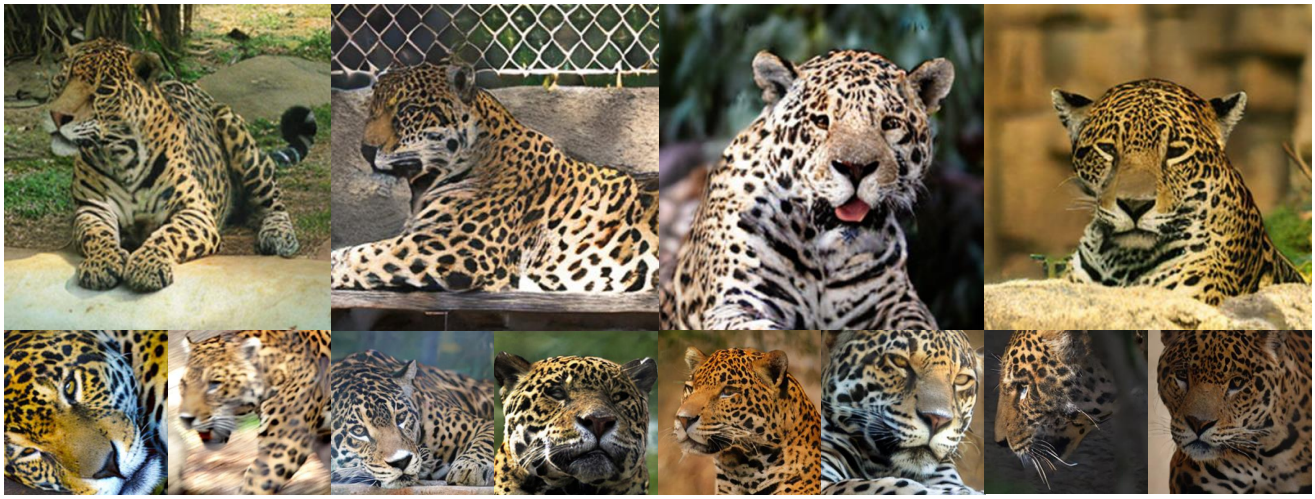

Figure 14. Generated samples of SiT-L/2 trained with D^2C using a 50K dataset (CFG=1.5). Class label = ”jaguar”(290)

Figure 15. Generated samples of SiT-L/2 trained with D^2C using a 50K dataset (CFG=1.5). Class label = ”otter”(360)

Figure 16. Generated samples of SiT-L/2 trained with D^2C using a 50K dataset (CFG=1.5). Class label = ”lesser panda”(387)

Figure 17. Generated samples of SiT-L/2 trained with D^2C using a 50K dataset (CFG=1.5). Class label = ”panda”(388)

Figure 18. Generated samples of SiT-L/2 trained with D^2C using a 50K dataset (CFG=1.5). Class label = ”fire truck”(555)

Figure 19. Generated samples of SiT-L/2 trained with D^2C using a 50K dataset (CFG=1.5). Class label = ”cheeseburger”(933)

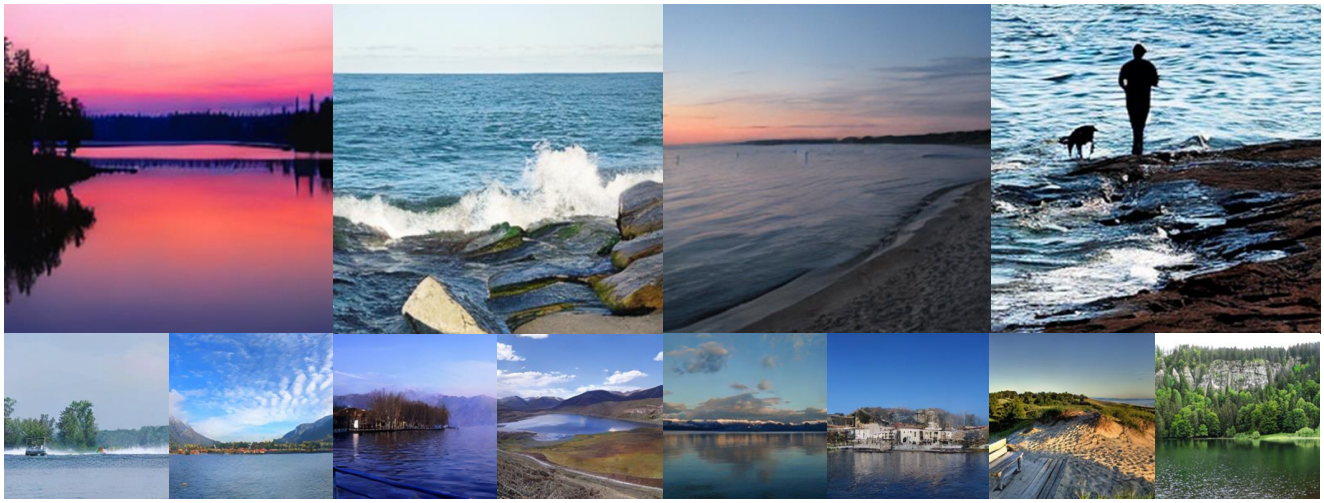

Figure 20. Generated samples of SiT-L/2 trained with D^2C using a 50K dataset (CFG=1.5). Class label = ”lake shore”(975)

Figure 21. Generated samples of SiT-L/2 trained with D^2C using a 50K dataset (CFG=1.5). Class label = ”volcano”(980)