EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Mathematisch-Naturwissenschaftliche Fakultät

Lernbasierte Computer Vision

Research Project

Dataset Distillation for Autonomous Driving

Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik
Lernbasierte Computer Vision
Zhengyu Su, zhengyu.su@student.uni-tuebingen.de, 2024

Bearbeitungszeitraum: 01.05.2024-bis 28.11.2024

Betreuer/Gutachter: Prof. Dr. Andreas Geiger, Universität Tübingen

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Arbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Zhengyu Su (Matrikelnummer 5500097), August 1, 2025

Abstract

Large datasets drive success in computer vision tasks like image classification, object detection, and semantic segmentation. However, managing these large datasets is a challenge. Dataset distillation is a technique that synthesizes a smaller yet equally effective version of a large dataset. In particular, minimax diffusion, a state-of-the-art approach, uses a diffusion model to generate a smaller dataset that maintains the diversity and representativeness of samples. Using the NAVSIM autonomous driving framework, we test the feasibility of this technique on complex driving data, which involves high-resolution camera images, and has not been used in prior work on dataset distillation. We develop a simplified driving task and corresponding baseline compatible with minimax diffusion. We also adopt a custom noise schedule and various parameter-efficient fine-tuning techniques to enhance details and maximize performance. Our final distilled dataset with 20 images achieves 74.4% accuracy, compared to 90.8% with the full 10.7k image dataset and 70.3% with 20 randomly sampled images, showing the feasibility of this approach for challenging real-world tasks.

Acknowledgments

I would like to thank Prof. Dr. Andreas Geiger for his valuable feedback, which helped me gain a deeper understanding of the research questions. I am deeply grateful to my mentor, Kashyap Chitta, for his constant support and advice throughout the project. His expertise and guidance were crucial in shaping the direction and outcome of my work. I also wish to express my appreciation to Melanie for the insightful discussions and thoughtful suggestions. Thank you all for your invaluable feedback and guidance.

Contents

1 Introduction

In the field of Computer Vision, deep learning has achieved great success across a wide range of tasks such as image classification, object detection, image segmentation, etc. In practice, the size and quality of the dataset has direct impact to a deep learning model performance, which is why there has been a constant push in the community to collect as much data as possible. However, many questions arise: How much data is indeed necessary to have a model trained with satisfactory performance? Does a model really need to process every single data sample in order to achieve optimal performance? Are there any data samples that contain redundant or repetitive information, which could be omitted without sacrificing the model performance? If we could remove or compress unnecessary data, significant computational resources and training time could be saved.

These questions and considerations lead to the research area known as Dataset distillation [WZTE18]. Dataset distillation focuses on techniques that aim to reduce large datasets into much smaller, synthetic datasets, while preserving performance on downstream tasks such as image classification. Rather than using a vast number of images, distilled datasets contain only a fraction of the original data. They are generated in a way that retains the key features required for effective model training. For example, instead of using thousands of images per class, dataset distillation generates a small set of images for each class. These synthetic datasets, often called "distilled datasets", are then tested on the same real validation sets to evaluate their performance compared to the full dataset.

While dataset distillation has shown promising results in various scenarios, its utility in more complex real-world tasks, such as autonomous driving, remains largely unexplored. Unlike classical image datasets, autonomous driving has a great need for datasets of various modalities: combining camera data with LiDAR sensor, Bird's Eye View map, ego status, bounding box annotations, and so on. They tend to be significantly larger in both resolution and dataset size, involving high-definition images and video streams. Handling and processing these datasets introduces challenges in terms of storage, computational resources, and time costs. In this context, we aim to apply dataset distillation in driving datasets that can make use of these different modalities efficiently. By compressing the data while preserving features needed for reasonable decision-making, dataset distillation can potentially accelerate the development of autonomous driving technologies. In our work, we propose a framework to deploy dataset distillation on an autonomous

vehicle dataset benchmark named "NAVSIM" [DHL+24] with a simplified setting of training a self-driving agent.

We formulate the driving task as a binary classification problem of determining the "brake" command based on front-view camera RGB image inputs. The labels "brake" and "not brake" are derived from the ego vehicle's acceleration through a predefined threshold. We explore how different fine-tuning techniques, loss functions, and noise schedules influence the quality of distilled driving datasets generated using the ImageNet pre-trained Diffusion Transformer (DiT) [PX23] as a foundation. This work represents a preliminary exploration into applying dataset distillation to complex driving data. Our distilled dataset of 20 images achieves 74.4% accuracy, which, although lower than the 90.8% accuracy from training on the full 10.7k image dataset, marks an improvement over the 70.3% accuracy obtained with 20 randomly sampled images. These findings provide a baseline and demonstrate the feasibility of applying dataset distillation to complex driving scenarios, highlighting the potential for further optimization through the integration of additional modalities.

In summary, while dataset distillation has proven efficient in traditional computer vision tasks, its application to the complex, multi-modal data in autonomous driving remains a relatively untapped area. The potential benefits, including reduced training times, lower computational costs, and improved efficiency, make it a promising direction for future research. In the following sections, we will explore how dataset distillation techniques can be applied to driving datasets and discuss the challenges and considerations that arise when working with such data.

2 Related Work

Dataset distillation is a growing area of research that has obtained significant attention in recent years. Numerous methods have been developed, and various applications have emerged. In this chapter, we review some of the foundational dataset distillation techniques that have been widely adopted. Additionally, we introduce basic autonomous driving agents, highlighting the intersection between dataset distillation and its potential applications in autonomous driving systems.

2.1 Autonomous Driving Datasets

Autonomous driving systems rely heavily on diverse and enriched datasets to address a wide range of driving scenarios. These datasets generally fall into two categories: synthetic datasets generated by simulators and real-world datasets captured using sensors such as cameras, LiDARs, and IMUs.

Early benchmark datasets like KITTI [GLU12] and Cityscapes [COR+16] provide high-quality image data with label annotations for tasks like object detection and semantic segmentation. However, their data is largely limited to German urban cities, restricting their diversity and scale. In contrast, the nuScenes [CBL+19] dataset expands this scope by introducing more complex scenarios across various continents, weather conditions, and vehicle types. nuScenes also offers a multi-modal setup, combining data from cameras, LiDARs, and radars. Building upon this, nuPlan [HC21] focuses on end-to-end ego-vehicle planning tasks. It includes both real-world data and a simulator, which allows for the testing and evaluation of planning algorithms. Meanwhile, OpenScene [Con23] serves as a compact redistribution of nuPlan, preserving essential annotations and sensor data while reducing the dataset size by over ten times, covering more than 120 hours of driving data from cities like Boston and Singapore, where complex driving environment often occur. NAVSIM [DHL+24] further extends this by offering a middle ground between traditional evaluation methods, combining real-world data with a non-reactive simulator. This approach collects simulation-based metrics, such as progress and time-to-collision, through short-horizon bird's-eye view simulations. NAVSIM's splits subsample the OpenScene dataset, with `navtrain` and `navtest`. This framework supports large-scale real-world benchmarking in autonomous driving research.

2.2 Knowledge Distillation

Knowledge distillation refers to the process of compressing the knowledge from a large, complex model (or an ensemble of models) into a smaller, more efficient model without a significant loss in performance [BCNM06][HVD15]. The smaller model, known as the "student" model, learns to mimic the output behavior of the larger model, referred to as the "teacher" model. Using ensemble learning and achieving the result by averaging the ensemble outputs is a common approach to improve a machine learning task performance [Die00]. By distilling the ensemble's knowledge into a single model, one can significantly reduce computational costs while retaining similar predictive capabilities.

Dataset distillation, while related, focuses on compressing the dataset itself rather than the model. Its goal is to create a smaller surrogate dataset that retains the essential information from the original, larger dataset. This approach seeks to reduce the number of images or data points needed for specific tasks without compromising performance. In the context of driving datasets, we aim to explore how dataset distillation can help optimize data efficiency and task performance within the scope of autonomous driving systems.

2.3 Data Pruning

Data pruning, or core-set selection, is another technique to reduce dataset size by retaining only the most valuable or representative samples, while still ensuring the models trained on this subset has similar performance to training on the entire dataset. Finding the best subset of a dataset is considered as an NP-hard optimization problem [MBL20], as given a dataset V, evaluating all possible subsets is of size $2^{|V|}$, which is computationally intractable, especially for large datasets. Therefore, most core-set selection methods are heuristic-based. For example, Herding [Wel09] chooses samples that are closest to the cluster center of each class and adds them incrementally to represent each cluster center. To address the issue of catastrophic forgetting, where models tend to forget patterns learned from older data when trained on new data [Fre99], the forgetting method [TSdC$^+$19] selects examples that exhibit the highest frequency of being forgotten during training to form a subset. This strategy aims to preserve information from earlier data by focusing on the most challenging examples. Some other methods include selecting the examples by finding the hard examples that the model is least confident about [CYM$^+$20].

Core-set selection methods often struggle with limited expressivity since the subsets are directly extracted from the original dataset. This constraint reduces their flexibility and ability to capture complex data patterns [AH23]. Additionally, studies have shown that although having better performance using these heuristic-based core-set methods in specific experimental settings, they can not outperform random selection in practical scenarios [GZB22].

2.4 Dataset Distillation

Dataset distillation and core-set selection both aim to condense a large dataset into a smaller, representative one. Some objectives $\mathcal{L}$ applied in core-set selection can also be used in dataset distillation, such as matching distributions and matching gradients. However, they differ significantly: core-set selection involves choosing samples directly from the original dataset, while dataset distillation generates synthetic data based on learned features from the original dataset. This distinction makes dataset distillation more scalable, as it doesn't rely on retaining raw data and can generalize more flexibly.

The dataset distillation framework involves two key steps: extracting a smaller synthetic dataset from the original large dataset and utilizing it to evaluate performance on downstream tasks. Different dataset distillation methods differ in the design of their optimization objectives, which can be categorized into three main approaches: performance matching, parameter matching, and distribution matching.

2.4.1 Performance Matching

The concept of performance matching was proposed in the first dataset distillation paper [WZTE18]. The synthetic dataset is treated as hyperparameters that need to be optimized in a bi-level fashion: the inner loop contains training the model on the synthetic dataset until the best fit, and the outer loop updates the synthetic dataset such that the model generalizes well on the real dataset. This approach works well, but with high computational demand due to the need of unrolling the inner loops to compute the gradient of validation loss through backpropagation of the entire network. Later work [DR22] improves the performance by adopting momentum-based optimizers and unrolling longer inner-loop.

To overcome the computational costs of bi-level optimization, Kernel Ridge Regression-based methods can be applied [NCL21]. These methods give a closed-form solution of finding distilled datasets by convex optimization. Instead of iteratively optimizing the neural network through backpropagation of gradients, the inner loop is replaced with a kernel model, which solves the bi-level problem. Based on this, [ZNB22] replace the kernel with a more flexible one with neural features to improve performance and help prevent overfitting.[LHAR22] use a random feature approximation kernel to reduce the complexity of matrix computation.

The above methods obtain the distilled dataset by directly optimizing the model performance. Another category of dataset distillation achieves comparable model performance by comparing the model parameters or gradients on both the original dataset and the synthetic one via defined objectives.

2.4.2 Parameter Matching

The core idea of parameter matching is to train the same network on the original dataset and synthetic dataset for some steps, such that the network parameters are approximately close $\theta^{\mathcal{S}} \approx \theta^{\mathcal{T}}$. First proposed in [ZMB21], gradient matching performs a single-step gradient update on the network using the synthetic dataset $\mathcal{S}$ and compares the gradient obtained from the original dataset $\mathcal{T}$, such that the distance (typically defined as cosine similarity) between these two gradients are minimized. After matching the gradients, the synthetic data is updated and subsequently used to train the network for T steps. This method is more memory-efficient than performance matching methods, as it requires only a one-step gradient computation for the update.

Many other methods build upon this foundation and have developed various extensions. Differentiable Siamese Augmentation (DSA) [ZB21] enhances gradient matching by applying data augmentation techniques to both synthetic and original datasets. These operations including flipping, cropping, rotating, etc. can be combined with any dataset distillation method since they are applied directly to the images. However, due to the unique characteristics of the driving dataset, we do not implement DSA in our experiments, as flipping vehicles or rotating scenes would not be appropriate. Information-intensive Dataset Condensation (IDC) [KKO+22] utilizes multi-representation as data parametrization to minimize the memory budget for storing condensed datasets. For example, images are stored at a lower resolution and upscaled as needed during training or evaluation, making it feasible to condense large datasets like ImageNet, where IDC serves as a baseline. IDC also enhances gradient matching by addressing the vanishing gradient problem caused by the small synthetic datasets in the inner loop. Instead, the synthetic data is optimized with networks trained on the original dataset, which regularizes the optimization.

Matching training trajectories [CWT+22] is another parameter matching method that improves the single-step approach by matching multiple steps on the distilled dataset. This approach overcomes the issue of error accumulation in the single-step method. Training trajectories are sequences of the network parameter updates over a timestep T: $\theta_{t=0}^{T-1}$. The expert trajectories obtained from the training process on the full original dataset guide the distilled dataset, ensuring that the training trajectory of a neural network on the distilled dataset closely resembles that on the original dataset. Similar to the performance matching methods, it faces the memory efficiency problem for unrolling the computational graph. TrajEctory matching with Soft Label Assignment (TESLA) [CWSH23] tackles this by calculating gradients for both the current network parameters and the synthetic datasets at each iteration. Additionally, it introduces soft label assignment for the synthetic dataset, which enhances convergence, particularly for datasets with a large number of classes.

2.4.3 Distribution Matching

Distribution matching [ZB22a] directly matches the distributions of the original and synthetic datasets in a latent space using an encoder (i.e. neural network), turning bi-level optimization in parameter matching methods into a single optimization. The distance between the distributions is measured using maximum mean discrepancy (MMD), and the process involves only tuning the learning rate. This makes the method faster and more efficient compared to other approaches.

2.5 Generative Dataset Distillation

With advancements in Generative Models, particularly models that operate in latent spaces such as GANs (Generative Adversarial Networks) and VAEs (Variational Autoencoders), the ability to generate high-quality, high-resolution images has greatly improved. These generative models open new doors for dataset distillation, allowing for more complex tasks and larger dataset scales to be tackled. Recent studies have explored the use of generative models in distillation tasks mainly using GANs and diffusion models.

Informative Training samples with GAN (IT-GAN) [ZB22b] uses a frozen pre-trained GAN to learn the informative latent code for generating informative samples. It differs from normal GAN [GPAM+14] that aims to generate samples that look real.

While IT-GAN distills dataset in pixel space, the more advanced method Generative Latent Distillation (GLaD) [CWT+23] conducts optimization in latent space, which helps prevent overfitting in the raw pixel space. A deep generative prior can be conveniently added into the distillation process, which significantly enhances the generalization of the synthetic dataset when training models that were not used during the distillation. Additionally, this approach allows the distillation of larger-resolution images while minimizing artifacts, such as high-frequency noise that occurs if distilled in pixel space.

Due to the intrinsic structure of GANs, they require GAN inversion, a resource-heavy process that involves mapping a real image back to its latent space. Choosing the right latent code to balance expressiveness and realism is also a complex and ill-posed problem [CWT+23].

In contrast, diffusion models have demonstrated the ability to generate higher-quality and more scalable images than GANs [DN21]. Unlike GANs, which rely on two adversarial networks that are trained jointly to generate images, diffusion models are likelihood-based. They progressively learn to denoise random noise into real images by modeling the data distribution. Additionally, diffusion models do not face the same challenges regarding the initialization of the latent code as GANs do. The latent code can be initialized simply using any pre-trained encoder, such as a VAE [KW22].

Classical diffusion models typically have a U-Net backbone [HJA20]. However, with recent developments in attention mechanisms and transformer architectures [VSP+23], a more advanced diffusion model known as the "Diffusion Transformer (DiT)" replaces the U-Net backbone with attention blocks [PX23]. Leveraging the power of the attention mechanism, this model family captures complex patterns in high-resolution images, enabling the generation of high-quality outputs. This model is also the state-of-the-art diffusion model family for class-conditioned image generation.

3 Method

This chapter presents the complete pipeline for applying dataset distillation to autonomous driving datasets. We outline the experimental setup in detail, including specific configurations and baseline models used for evaluating the effectiveness of dataset distillation in this context. As no prior work has explored dataset distillation for driving data, we establish these baselines to facilitate meaningful comparisons and provide a foundation for future research in this domain.

3.1 Preliminaries

3.1.1 Dataset: NAVSIM

Before going into the methods in detail, we describe the dataset used in this work. NAVSIM is a simulation-based driving framework that generates scenes over a short time horizon [DHL+24]. It utilizes pre-defined data splits for agent training and evaluation. This study uses the `navtrain` split as the training dataset to train a simplified binary classifier agent that predicts whether to apply the brake command. The corresponding validation dataset is the `navtest` split. Both datasets are derived from the OpenScene dataset [DHL+24], [Con23], [STW+23], [YCSL23].

Each scene in the dataset consists of metadata containing a sequence of history and future frames. Each frame includes attributes that are useful for training an agent, such as timestamps, traffic lights, bounding box annotations, LiDAR information, ego vehicle status, and camera sensor captures. However, since our focus is on exploring image data distillation, we restrict our training to the RGB images captured by the front-facing camera, simulating the driver's view. Additionally, we use longitudinal acceleration data to determine the "brake" or "no brake" command. All other information is discarded during training. To fully control a vehicle, however, other information may come into use to control the steering. In the next subsection, we discuss the implementation of a driving agent based on a brake/no brake classifier.

3.1.2 Driving Agent

Since dataset distillation is primarily applied in classification tasks on datasets like ImageNet and CIFAR, there has been limited experimentation with driving datasets. Therefore, our first step is establishing a baseline for driving datasets, which will

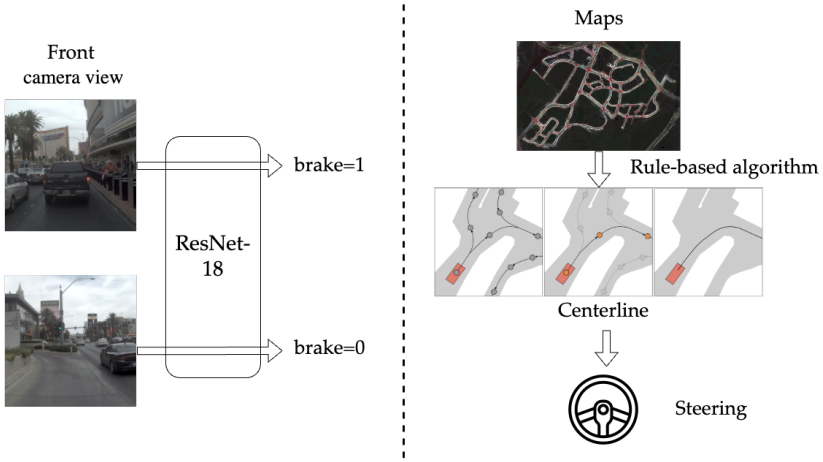


Figure 3.1: Illustration of the whole vehicle control. Left: image-based binary classifier for throttle and brake command. Right: Acquisition of steering command based on a map and a rule-based algorithm for route planning.

later serve as a foundation for evaluating the distilled dataset. To achieve this, we simplify the NAVSIM dataset and define the driving agent as a vision-based binary classifier (see Figure3.1), tasked with predicting whether to brake or not based on the front camera view.

We use ResNet-18 as the model architecture for our baseline classifier. We initialize it with an ImageNet pre-trained model checkpoint and fine-tune it on the NAVSIM dataset to recognize brake commands. With this agent, we gain control over the vehicle's speed based on its predictions. Further details on dataset pre-processing and the experimental setup are provided in Section 4.2 To fully control the vehicle, we also need to manage its heading angle, or in other words, its position. Since map data is provided in the metadata of the NAVSIM framework, it is possible to extract a centerline from the map using a rule-based graph search algorithm, given both a starting point and a target destination. This centerline provides the location of the vehicle within the map [DHGC23]. By integrating speed control and positional information, we can achieve full control of the vehicle.

3.1.3 Dataset Distillation

Given a large-scale dataset $\mathcal{T} = \{(\mathbf{x_i}, y_i)\}_{i=1}^{N_T}$ with N_T being that dataset size, each data sample $\mathbf{x_i}$ is labeled with y_i. The goal of dataset distillation, also known as dataset condensation, is to find a smaller surrogate dataset $\mathcal{S} = \{(\mathbf{x_i}, y_i)\}_{i=1}^{N_S}$ with the size of N_S being much smaller than the original dataset size N_T (i.e. $N_S \ll N_T$). Despite the smaller dataset size, the surrogate dataset $\mathcal{S}$ should achieve a similar validation performance evaluated using the same metrics as the original dataset $\mathcal{T}$.

Formally, dataset distillation can be defined as finding the optimal surrogate dataset $\mathcal{S}$ that minimizes the loss $\mathcal{L}(\mathcal{S},\mathcal{T})$, where $\mathcal{L}$ is the objective function for dataset distillation:

$$\mathcal{S} = \arg\min_{\mathcal{S}} \mathcal{L}(\mathcal{S},\mathcal{T}) \tag{3.1}$$

3.2 Dataset Distillation via Diffusion

Diffusion models can reconstruct a dataset distribution through its forward noising and backward denoising processes. Given a data point sampled from a given data distribution $\mathbf{x}_0 \sim q(\mathbf{x}_0)$, noise defined by a schedule is added at each timestep t during the forward process, until a total number of T diffusion steps. This noise schedule controls the mean and variance of the noise added to the previous timestep. At each timestep a version of the input sample data with a certain amount of noise is generated as $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$. These samples are generated iteratively and follow the Markov assumption, meaning that at each timestep, the sample is affected only by the previous timestep. For each timestep $t \in \{1, \ldots, T\}$, the following equation holds:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \tag{3.2}$$

with $\beta_t \in (0,1)$ being the variance of the noise at timestep t.

As the timestep becomes larger, $T \longrightarrow \infty$ and the last $\mathbf{x}_T$ will be just as a Gaussian distribution. Following the Markov property, it can be written as follows:

$$q(\mathbf{x}_1, \ldots, \mathbf{x}_T|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \tag{3.3}$$

There are two helper parameters defined: $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t :- \prod_{i=1}^{t} \alpha_i$. They are associated with how much of the original data remains in the noisy data after each timestep. We can compute $\mathbf{x}_t$ by sampling from the distribution in equation 3.2 from arbitrary timestep with the reparameterization trick [KSW15]:

$$\mathbf{x}_t = \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\epsilon \text{ with } \epsilon \sim \mathcal{N}(0, \mathbf{I}) \tag{3.4}$$

$$= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\epsilon \tag{3.5}$$

$$\overset{(3.2)}{=} \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon \tag{3.6}$$

and

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, 1-\bar{\alpha}_t\mathbf{I}) \tag{3.7}$$

Now we can describe the noise added to the original input sample $\mathbf{x}_0$ to get the noisy data at any timestep t by the normal distribution with the mean of $\sqrt{\bar{\alpha}_t}$ and the variance $1-\bar{\alpha}_t$. This allows a direct computation from any arbitrary timestep and it helps avoid the complex process of iteratively adding noise at every single timestep.

The backward denoising process is characterized by a neural network with parameters θ that learns the mean and variance of the noise. It reconstructs the original clean data by iteratively subtracting the predicted noise from the noisy data $\mathbf{x_T}$, which follows a Gaussian distribution $p(\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_t; 0, \mathbf{I})$. The denoising process can be written as follows:

$$p(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \text{ with } p(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_\theta(\mathbf{x}_t, t)) \tag{3.8}$$

3.2.1 Model Training

During training, the negative log-likelihood of the data learned under the diffusion model distribution is approximated through the variational lower bound and the diffusion loss is defined as the expectation of the variational lower bound. In practical implementation, diffusion loss is often simplified as the mean squared error (MSE) between the predicted noise and the actual noise, as predicting noise with the simplified error performs better than predicting the mean with a fixed variance using the theoretical variational lower bound [HJA20].

For latent diffusion models, the sampled data $\mathbf{x}_0$ is first passed through an encoder E to generate a latent code $\mathbf{z}_0 = E(\mathbf{x}_0)$. The entire diffusion training process is also performed in the latent space. The reconstructed images $\hat{\mathbf{x}}$ need to be decoded by a decoder D corresponding to the latent encoder to convert back to the image space.

3.2.2 Classifier-Free Guidance (CFG) and Image Generation

After fine-tuning the model to the defined dataset, we use the class condition to generate a certain number of images for each class (images per class, IPC). This is done by random sampling several latent codes $\mathbf{z}_T \sim \mathcal{N}(0, \mathbf{I})$ from a normal distribution, and the labels are set with classifier-free guidance to balance between sample quality and diversity [HS22] by combining guided generation (with a true label) and unguided generation (with a null label). Following the backward denoising process, latent codes at timestep 0 are decoded back to the image space, the resulting outputs are then the generated images.

3.3 Minimax Diffusion

Dataset distillation aims to generate a surrogate dataset that accurately represents the original dataset and comprehensively covers the underlying data distribution. This dual objective of representativeness and diversity is addressed in the Minimax Diffusion Dataset Distillation paper [GVK+24]. The authors introduce two minimax

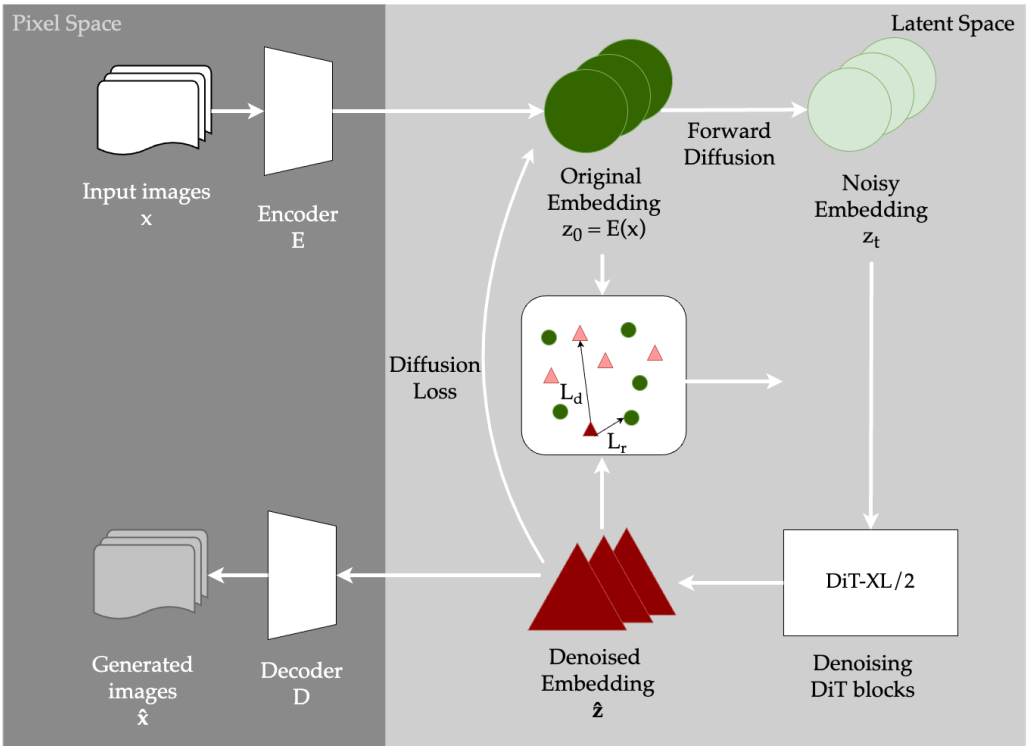


Figure 3.2: Minimax Diffusion pipeline

criteria which are incorporated as additional loss terms alongside the standard diffusion loss, resulting in a combined loss function for training. This approach integrates seamlessly into any existing training loops, making it easy to implement. Figure 3.2 shows the overall pipeline of the minimax diffusion method, illustrating how the two key criteria are calculated to the current prediction. The following sections provide a detailed description and formal definitions of these criteria.

3.3.1 Simple Diffusion Loss

The diffusion loss is measured between the predicted noise based on a noisy latent code $\mathbf{z}_t$ and a class condition label $\mathbf{c}$. This controls the class labels of the diffusion process and can restrict the class of generated images (guided diffusion). Given the true noise ϵ, the diffusion loss is formulated as

$$L_{\text{simple}} = \|\epsilon_\theta(\mathbf{z}_t, \mathbf{c}) - \epsilon\|^2 \tag{3.9}$$

3.3.2 Representativeness

Representativeness describes how well the surrogate dataset captures the characteristics of the original data. The initial approach focuses on minimizing the difference

between the embedding distributions of the surrogate and original datasets by maximizing the cosine similarity between them:

$$\mathcal{L}_r = \arg\max_{\theta} \sigma\left(\hat{\mathbf{z}}_\theta(\mathbf{z}_t, \mathbf{c}), \frac{1}{N_B}\sum_{i=0}^{N_B} \mathbf{z}_i\right) \tag{3.10}$$

In this equation, $\hat{\mathbf{z}}_\theta(\mathbf{z}_t, \mathbf{c}) = \mathbf{z}_t - \epsilon_\theta(\mathbf{z}_t, \mathbf{c})$ represents the predicted original embedding by subtracting the noise ϵ_θ from the noisy embedding $\mathbf{z}_t$, N_B denotes the mini-batch size of the real samples. However, the simple alignment method often results in the predicted embeddings being drawn toward the center of the original distribution, thereby weakening its diversity.

To address this issue, the minimax criterion for representativeness is proposed. This criterion utilizes an auxiliary memory $\mathcal{M} = \{\mathbf{z}_m\}_{m=1}^{N_M}$ with memory size $|\mathcal{M}| = N_m$. This memory stores real samples from adjacent iterations. In this way, only a small subset of the real samples are maintained and the learned embeddings are prevented from becoming overly concentrated in the center of the real distribution. The objective is formulated as follows:

$$\mathcal{L}_r = \arg\max_{\theta} \min_{m \in [N_M]} \sigma(\hat{\mathbf{z}}_\theta(\mathbf{z}_t, \mathbf{c}), \mathbf{z}_m) \tag{3.11}$$

This criterion encourages the predicted embedding to be as close as possible to the real embedding that is least similar to it (i.e., with the lowest cosine similarity) so that the predicted embedding can also cover samples that are not centered within the original distribution. Consequently, the predicted embedding remains representative of samples located at the edges. The diffusion objective serves to draw the predictions towards the original distribution.

3.3.3 Diversity

The second minimax criterion focuses on diversity. It encourages the predictions to be diverse from each other yet still represent the original dataset. This can be considered as identifying representative samples from various subset clusters that form a larger dataset. An additional auxiliary memory $\mathcal{D} = \{\mathbf{z}_d\}_{d=1}^{N_D}$ is employed to store the predicted embeddings from adjacent iterations. In contrast to the representativeness criterion, this criterion focuses on the predictions and pushes the current prediction away from the most similar prediction stored in auxiliary memory (i.e., the one with maximal cosine similarity). The objective is defined as:

$$\mathcal{L}_d = \arg\min_{\theta} \max_{d \in [N_D]} \sigma(\hat{\mathbf{z}}_\theta(\mathbf{z}_t, \mathbf{c}), \mathbf{z}_d) \tag{3.12}$$

It is essential to note that maintaining a balance between representativeness and diversity is critical. Focusing more on representativeness may lead to the neglect of edge cases, whereas too much diversity can compromise representativeness.

Combining all three objectives, the final objective is the minimax diffusion distillation objective:

$$\mathcal{L} = \mathcal{L}_{\text{simple}} + \lambda_r \mathcal{L}_r + \lambda_d \mathcal{L}_d \tag{3.13}$$

where λ_r and λ_d are hyperparameters that control the weights of the two minimax criteria.

In summary, the input image batch $\mathbf{x}$ is first transformed from pixel space to latent space through an encoder E. The output of this encoder yields the original embeddings $\mathbf{z}_0 = E(\mathbf{x})$. During the forward pass, Gaussian random noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is added to obtain the noisy embedding $\mathbf{z}_t = \sqrt{\bar{\alpha}_t}\mathbf{z}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$. The diffusion model employed in this work is a DiT-XL/2 model, pre-trained on the ImageNet dataset at a resolution of 256×256 pixels. These DiT blocks learn the embedding distribution using the combined loss as previously described, and they predict the noise $\epsilon_\theta(\mathbf{z}_t, \mathbf{c})$ from the noisy embedding $\mathbf{z}_t$ and the class condition $\mathbf{c}$. By subtracting the noise from the noisy embedding, we obtain the estimated embedding prediction $\hat{\mathbf{z}}_\theta(\mathbf{z}_t - \epsilon_\theta(\mathbf{z}_t, \mathbf{c}))$. After each iteration, both the model parameters and the auxiliary memories are updated. Specifically, the auxiliary memories are implemented as FIFO queues, where the current real embedding is enqueued into $\mathcal{M}$, and the current estimation of the embedding will be enqueued into $\mathcal{D}$. The oldest samples in each memory are discarded according to the specified size parameters. We also discuss the selection of tunable parameters in section 4.3.3.

3.4 Fine-tuning Techniques

DiffFit. By fine-tuning only the bias terms and newly added scaling factors in specific layers of the diffusion model, DiffFit achieves significant efficiency gains. Compared to full fine-tuning approaches, it requires less than 1% of the model parameters to be fine-tuned, while the training speed is 2 times faster [XYS+23].

During DiffFit fine-tuning, most of the model weights are frozen, only bias terms, layer normalizations, y-Embedder for the class embedding, and the added trainable scale factor γ in the attention blocks are updated. In the DiT-XL/2 model for 256×256 resolution input images, the number of DiffFit tunable parameters is 1.74M.

Low-Rank Adaptation (LoRA). LoRA is another method designed to make fine-tuning large pre-trained models more efficient by reducing the number of trainable parameters. As illustrated in Figure 3.3, this is achieved by decomposing the weight updates into low-rank matrices, which allows for modifying the model's behavior with fewer computational resources and lower storage use [HSW+21]. The authors of the LoRA paper hypothesize that when adapting the weights of a dense layer to a specific task, the updates of the weights should have a low "intrinsic rank" [HSW+21]. Given a weight matrix $W_0 \in \mathbb{R}^{r \times k}$ of a pre-trained model, its update matrix ΔW will be decomposed to two low-rank matrices $\Delta W = BA$ where $B \in \mathbb{R}^{d \times r}$, $A \in \mathbb{R}^{r \times k}$ and $r \ll \min\{d, k\}$. The pre-trained weight matrix W_0 remains untouched during training,

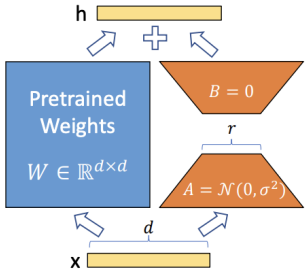


Figure 3.3: Illustration of decomposing pre-trained weights with two low-rank matrices A and B. [HSW+21]

only the parameters in the two low-rank matrices A and B are trainable. At the beginning of the training, A is initialized with a normal distribution $\mathcal{N}(0, \sigma^2)$ and B is a zero matrix, so to make sure that at the beginning there will be no major change in the weight parameters. In the LoRA configuration, setting a higher r value means tuning more parameters. Apart from rank r, another tunable hyper-parameter α controls how much of the low-rank matrix will be added to the updates by scaling $\frac{\alpha}{r} w_0$. We discuss the hyper-parameter settings in the next Chapter.

In the transformer-based DiT model we use, we apply LoRA to the weights of the query, key, value pairs, and the projection (W_q, W_k, W_v, W_p) of the self-attention blocks.

3.5 Noise Schedule

In the Minimax Diffusion paper, the authors conducted experiments within the ImageNet dataset. Due to the nature of such datasets, which typically contain single, large, dominant objects, the classical linear noise schedule is sufficient. However, as noted in another study that experimented with diffusion models on driving data [TJZ+24], the classical linear schedule may not be the most effective for these scenarios. This is because driving datasets contain small but critical objects, such as cars, pedestrians, and traffic lights, which are essential for making driving decisions. These objects often consist of only a small number of pixels, therefore quickly losing their shape and detail in the early stages of the forward process when a linear noise schedule is applied. In driving scenarios, these smaller objects are vital for safety and must be carefully preserved.

To address this, the authors of the DriveDitFit paper [TJZ+24] proposed a Spoon-Cosine noise schedule, named after its β_t curve, which resembles the shape of a spoon (see Figure 3.4). Precisely, this schedule employs a cosine-like noise schedule in the early stages of the diffusion process to preserve small details within the image. As the process progresses, it gradually shifts towards a linear schedule in the later stages, ensuring that the mapping between the Gaussian distribution and data

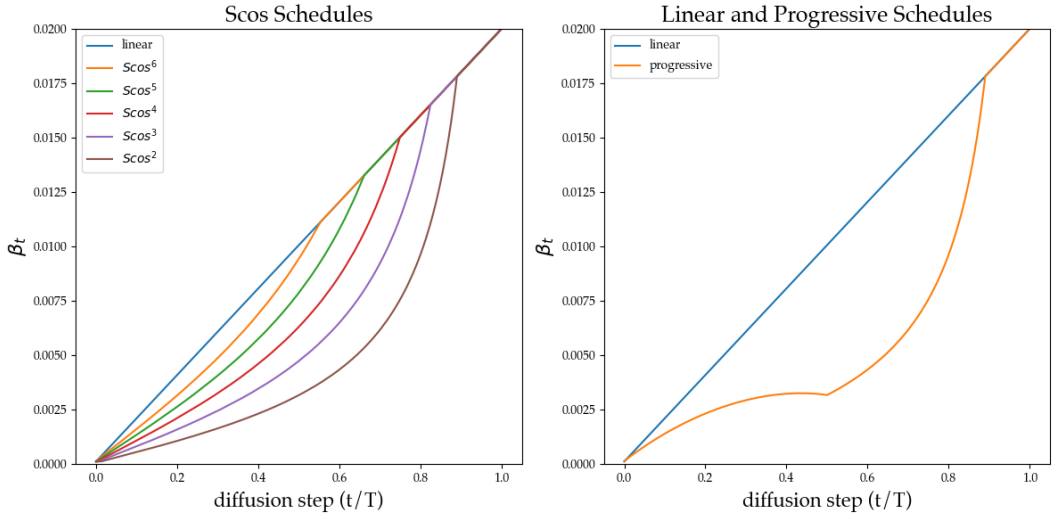
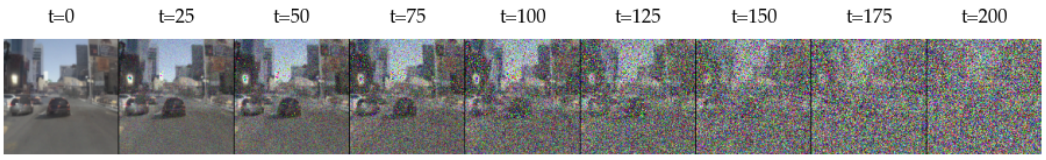


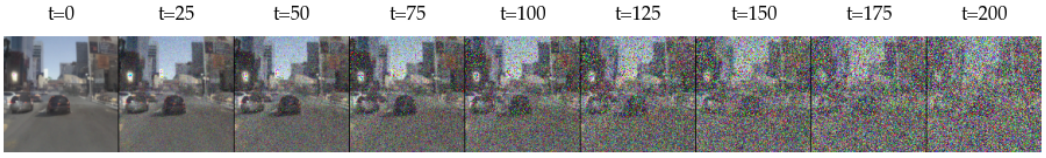
Figure 3.4: β_t curve for spoon-cos schedules and the progressive tuning from $Scos^6$ to $Scos^2$.

distribution learned by the pre-trained diffusion model is not damaged. The authors also describe a progressive tuning process, which involves transitioning from an $Scos^6$ schedule to an $Scos^2$ schedule at a certain time interval τ during fine-tuning. We adopt both this noise schedule and the associated tuning method in our experiments.

A comparison between the linear and cosine noise schedules illustrates the difference in a driving data sample. As shown in Figure3.5a and 3.5b, the first row (a) represents images affected by a Linear noise schedule at different time steps. Here, we observe that the small objects, such as cars and traffic lights, begin to blur and lose clarity as early as $t = 50$, and by $t = 100$, most of these features are almost indistinguishable from noise. In contrast, the second row (b) shows the same sequence of images with the Spoon Cosine noise schedule applied. Even at $t = 150$, small objects remain recognizable, highlighting the effectiveness of the cosine schedule in retaining crucial details over a longer period.



(a)



(b)

Figure 3.5: Comparison between the Linear and Spoon Cosine noise schedules on driving data. (a) Linear noise schedule, which leads to faster degradation of small features, and (b) Spoon Cosine noise schedule, which retains smaller details for longer during the noise process.

4 Experiments

This chapter summarizes the experiments conducted. We begin by testing the entire latent diffusion distillation training and deployment pipeline on the CIFAR-10 dataset, which provides a manageable setting to verify the effectiveness of the minimax diffusion method. We then extended our approach to the NAVSIM dataset for more complex, driving scenarios.

4.1 Minimax Diffusion Experiments on CIFAR-10

Dataset. The CIFAR-10 dataset is a human-labeled subset of the Tiny Images dataset [Kri09]. It consists of 60,000 samples in the following ten classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. There are 5000 training samples and 1000 test samples for each class. All the samples are colored images and are in 32×32 resolution.

Model. As the resolution of CIFAR-10 is low compared to the pre-trained DiT-XL/2 model, which is trained on 256×256 resolution images and used in minimax diffusion by default, we use a CIFAR-10-based diffusion model. We fine-tuned the CIFAR-10 pre-trained conditional diffusion model from [KAAL22] with the minimax criteria added to the training loop.

Results. After a hyper-parameter search, we find that setting $\lambda_r = 8, \lambda_d = 16$, memory size=64 with a constant learning rate of 0.001 using the Adam optimizer gives us the best validation performance. As shown in Table 4.1, after training for around 15.6k iterations with a global batch size of 256 on 4 GPUs, the validation accuracy is around 50.4% for 50 IPC, while the random selection of 50 IPC gives a validation accuracy of around 43.4%. Figure 4.1 shows some of the generated images of each class after minimax fine-tuning on the right side, highlighting subtle differences such as the perspective compared to the pre-trained diffusion model images.

Method	Validation accuracy
Random	43.4 ± 1.0
Pre-trained	47.3 ± 0.3
Minimax fine-tuned	50.4 ± 0.3

Table 4.1: **Minimax Diffusion on CIFAR-10**. Validation accuracy of a CNN classifier trained on 50 IPC for each method.

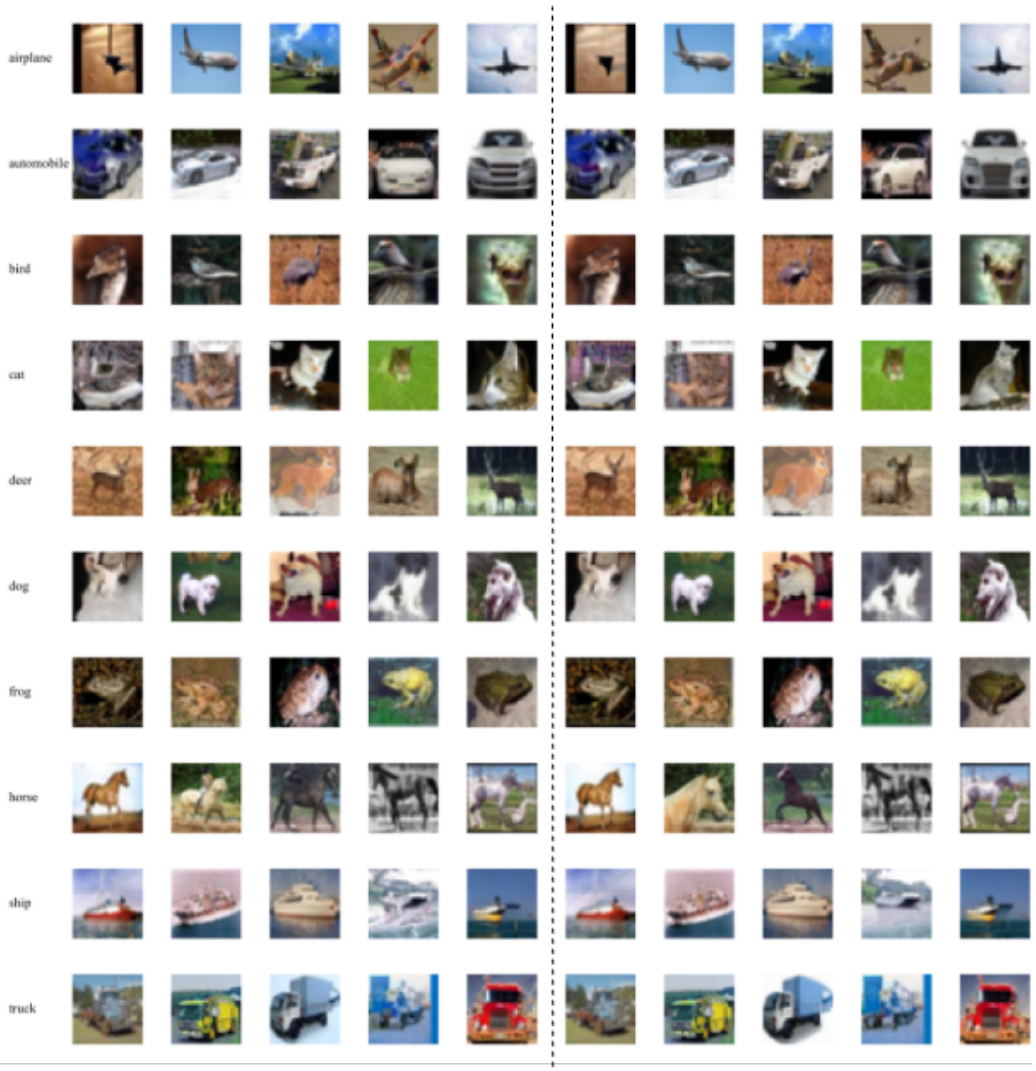


Figure 4.1: Example of generated CIFAR-10 images of the pre-trained diffusion model (left) and the minimax fine-tuned model (right).

4.2 Minimax Diffusion Experiments on Driving Data

4.2.1 The NAVSIM Framework

NAVSIM is a framework for non-reactive autonomous vehicle simulation [DHL+24]. We use the dataset splits provided by this framework `navtrain` and `navtest`, sampled from the largest 3D driving dataset OpenScene project [Con23] and filtered out the frames with trivial solutions and annotation errors. The filtered NAVSIM dataset consists of 85,109 training scenes and 18,179 validation scenes. These scenes are the agent inputs consisting of the current frame and three previous frames

with the RGB images in 1920×1080 pixel resolution each captured from one of the eight cameras surrounding the vehicle and a combined LiDAR point cloud from five sensors. Additionally, bounding box information and the ego status (velocity, acceleration, pose, navigation command) are also provided for each frame indicating the vehicle's situation [DHL+24].

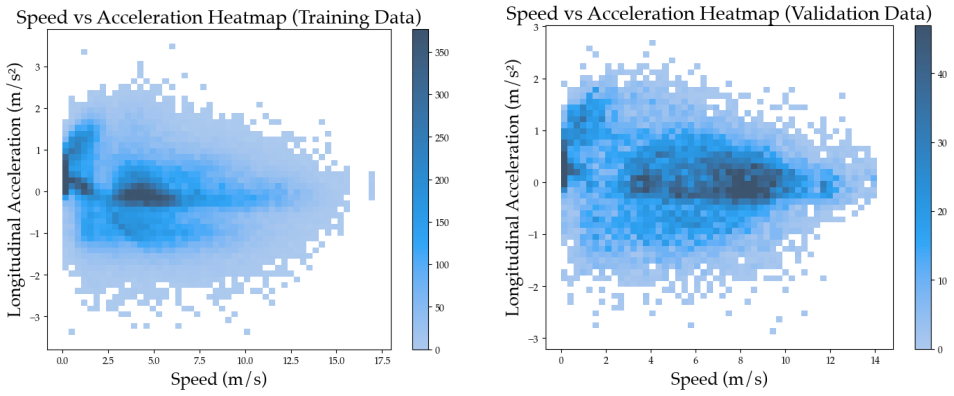
4.2.2 Task Formulation

To align with the downstream application of dataset distillation while maintaining a reasonable pipeline for deployment in the research field of autonomous driving, we formulate the agent's decision-making as a binary classification task: predicting whether to apply a brake or not, based solely on the front camera view. To ensure a balanced class distribution, we set the threshold for the brake command at $0\,\mathrm{m/s}$, resulting in 39,273 "no brake" samples and 45,836 "brake" samples in the training set, and 9,476 "no brake" samples and 8,703 "brake" samples in the validation set. In further discussion, we will focus on the "brake" command. Although more detailed classifications could be made —such as estimating the throttle command based on velocity and acceleration —other factors from different modalities also play a role. We avoid these complexities in our experiment and focus on establishing a simple baseline by determining the "brake" command solely from vehicle acceleration. This approach allows us to start with a straightforward model before expanding to more nuanced cases, which could be explored in future research.

4.2.3 Data Pre-processing

As previously mentioned, like a human driver, a driving agent's decision-making relies on multiple modalities. A complex agent must coordinate information from various sources, such as traffic lights, speed, and road conditions. However, since we aim to simplify the agent into a vision-based system, we need to clean the dataset to ensure that it is dependent on visual information and avoid scenarios where acceleration and velocity may also affect the agent's decision-making. The heatmap (shown in Figure 4.2) gives us an insight into driving patterns by showing the relationship between speed and longitudinal acceleration. Areas with high density (dark regions with low acceleration and very low speed) indicate vehicles with stable speeds and accelerations. This combination is often related to scenarios like slow traffic situations. We want to filter out these scenes and focus more on dynamic driving behaviors.
To achieve this, further filtering was applied to refine the dataset:

1. **Command-based filtering:** We filter out scenes where the navigation command is set to "turn left", "turn right", or "undefined", as these could distort the front camera view.



(a) (b)

Figure 4.2: Heatmap of speed vs. longitudinal acceleration. Note the high density near zero acceleration, indicating that vehicles frequently maintain a constant speed. Low density for high speeds and accelerations, indicating more dynamic driving behaviors.

2. **Acceleration-based filtering:** We remove scenes where the acceleration was very low (accelerations between $-1\,\mathrm{m/s^2}$ and $1\,\mathrm{m/s^2}$). Such scenes may misrepresent braking behaviors and do not represent typical driving scenarios in real-world driving, where braking decisions are more critical.

3. **Speed-based filtering:** We exclude scenes where the vehicle's speed was below $3\,\mathrm{m/s}$. As can be seen in Figure 4.2, low speeds often correlate with low acceleration values and may represent scenarios such as idling, congestion, or traffic stops. In these situations, braking decisions become less critical and may not accurately reflect typical driving behavior.

These filtering steps are essential to ensure that the dataset remains relevant to the task of predicting brake commands based on visual inputs and is more representative of active driving, minimizing the influence of non-visual factors.
After filtering, the camera images with an original resolution of 1080×1920 are cropped into a square shape of 1080×1080 by focusing on the center of the width. Following this, the images are downsized to a resolution of 256×256. We chose this method instead of directly resizing to 256×256 because we have found that this cropping process improves classification accuracies for this specific task. We assume that this approach mimics the way a human driver focuses their field of view at the center of the scene, thus enhancing the model's performance in understanding the most relevant visual information for decision-making.
In the end, the dataset we use for all the experiments consists of 10,714 training samples (with 2,859 "not brake" and 7,855 "brake") and 1,725 validation samples (with 682 "not brake" and 1,043 "brake").

4.3 Baseline Models

As the entire pipeline contains two major parts: distilling the original dataset into smaller surrogate data and deployment of the surrogate data on a downstream task, there are two model architectures involved. For distilling, the Diffusion Transformer (DiT) model, and for classification, the ResNet-18 model. Both are pre-trained on the ImageNet dataset and we perform fine-tuning strategies to map the data distribution to the driving dataset.

4.3.1 Diffusion Model for Distillation

Diffusion Transformers are a class of transformer-based latent diffusion models for high-resolution class-conditioned image generation with high quality. Classical diffusion models have U-Net as their backbones and Peebles et al [PX23] define DiT with a transformer as the backbone. In terms of FID, they achieve the lowest state-of-the-art of 2.27 on ImageNet 512×512 and 256×256 benchmarks with the largest model of its class DiT-XL/2. This is also the model we use as a distillation baseline.
The entire DiT-XL/2 model has 675M tunable parameters. For fine-tuning tasks, modifying all model parameters is often unnecessary. Only the task-relevant parameters, which make up a fraction of the entire model, need to be adjusted. This also speeds up fine-tuning and reduces resource consumption, as fewer parameters are updated. We use the following two parameter-efficient fine-tuning strategies to adapt the model parameters from the ImageNet domain to the driving dataset domain.

4.3.2 Binary Classifier as Driving Agent

After filtering the dataset, we train the baseline binary classifier for predicting brake commands. We conduct experiments on both 256×256 and 64×64 resolution images, using only RGB images captured by the front camera.

Training. We train with an AdamW optimizer with an initial learning rate of 1e-4 and a learning rate decay of 1e-4 for regularization. Additionally, we employed a ReduceLROnPlateau scheduler, which adjusts the learning rate based on the validation loss. All experiments ran for approximately 2,000 steps on a single 2080 GPU with a batch size of 256.

Results. The classifier achieves validation accuracies of approximately 84% for 64×64 images (Table 4.2) and 90.78% for 256×256 images (Table 4.2). We experiment with a lower resolution to speed up training time and also test at a higher 256×256 resolution to match the input size required for further experiments with the DiT model.

Filter Out	#Training samples	Resolution	Validation Accuracy
No filter	85,109	64×64	63.75
		256×256	71.40
$a \in [-1, 1]$	18,197	64×64	80.71
		256×256	88.74
$a \in [-1, 1] \wedge v < 3$	10,714	64×64	84.00
		256×256	**90.78**

Table 4.2: Effects of various data filtering strategies on the number of training samples and corresponding validation accuracies at different image resolutions (64×64 and 256×256). The first row shows results without filtering, while the second and third rows show the effects of filtering scenes where acceleration (a) is between $-1\,\mathrm{m/s^2}$ and $1\,\mathrm{m/s^2}$, and where velocity (v) is less than $3\,\mathrm{m/s}$.

4.3.3 DiT Fine-tuning on NAVSIM

During the fine-tuning phase, we conducted experiments using the ImageNet pre-trained DiT-XL/2 model at a resolution of 256×256 with constant learning rates. We experiment with two learning rate choices 1e-3 and 1e-4. With a batch size of 32 on a single A100 GPU, the generated images began to closely resemble the style of the NAVSIM driving dataset after approximately 20,000 training steps. After fine-tuning, the images are generated from random noise with 50 denoising steps. These settings will serve as our default configuration for subsequent experiments on distillation fine-tuning, ensuring consistency and reliability in our results.

Since the baseline agent is trained with a batch size of 256, it is crucial to set the number of IPC to generate. Given that our model only distinguishes between two classes, we set the IPC number to 250, ensuring no duplicates are being trained in the same batch.

Following this setting, the agent will be trained for 100 steps on the surrogate dataset with the same hyperparameter settings as the baseline agent trained on the original dataset.

After training, we evaluate the surrogate agent's performance through overall accuracy and class-specific accuracy, allowing for insights into its decision-making across different driving scenarios. The surrogate dataset derived from the DiffFit fine-tuned diffusion model serves as the baseline for these experiments. We further explore the impact of fine-tuning strategies like adjusting LoRA settings and noise schedules, as well as tuning minimax parameters to optimize the model's performance. In the following section, we conduct a detailed parameter and method analysis.

LoRA. Increasing the LoRA rank r enhances model adaptability by modifying more parameters. In our experiments, we applied LoRA to the pre-trained DiT model

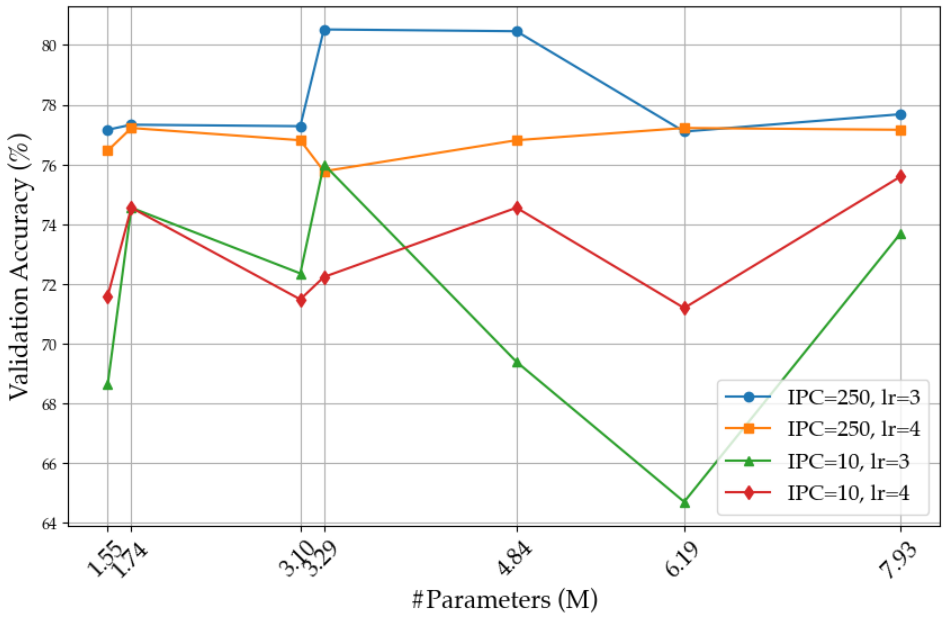


Figure 4.3: Validation Accuracy for Different IPC and Learning Rates with DiT fine-tuning using LoRA (1.55, 3.10, 6.19), DiffFit (1.74), and LoRA with DiffFit (3.29, 4.84, 7.93).

by injecting LoRA matrices into attention blocks. Figure 4.3 shows the validation accuracy for experiments with IPC 10 and IPC 250 across various LoRA settings combined with DiffFit fine-tuning. The x-axis indicates the number of modified parameters for each setting. During training, r and α are set equally. As suggested by the LoRA paper, tuning α acts similarly to adjusting the learning rate when using the Adam optimizer, which we employ in our case.

Specifically, LoRA with rank 8 modifies 1.55M parameters, or 0.23% of all tunable parameters. As the rank doubles, the modified parameters also double, with rank 16 3.10M and rank 32 6.19M parameters being affected. In configurations incorporating DiffFit, 3.29M, 4.84M, and 7.93M parameters are adjusted with the corresponding LoRA ranks (8, 16, and 32). In general, adding DiffFit to LoRA with a constant learning rate 1e-3 improves performance, since DiffFit modifies the y-embedder, which may contribute to better classification. A moderate LoRA rank combined with DiffFit yields the best overall performance, while higher ranks without DiffFit tend to perform worse. Figure 4.4 provides a more detailed view of class-specific validation accuracies, indicating that the "not brake" command is classified more accurately than the "brake" command. Given the critical importance of accurate "brake" classification for safety in autonomous driving, this class should be considered more. With a LoRA rank of 8 and DiffFit, the "brake" accuracy reaches 78.72% for IPC 250 and 72.10% for IPC 10, while the "not brake" accuracy is 86.07% for IPC 250 and 81.96% for IPC 10.

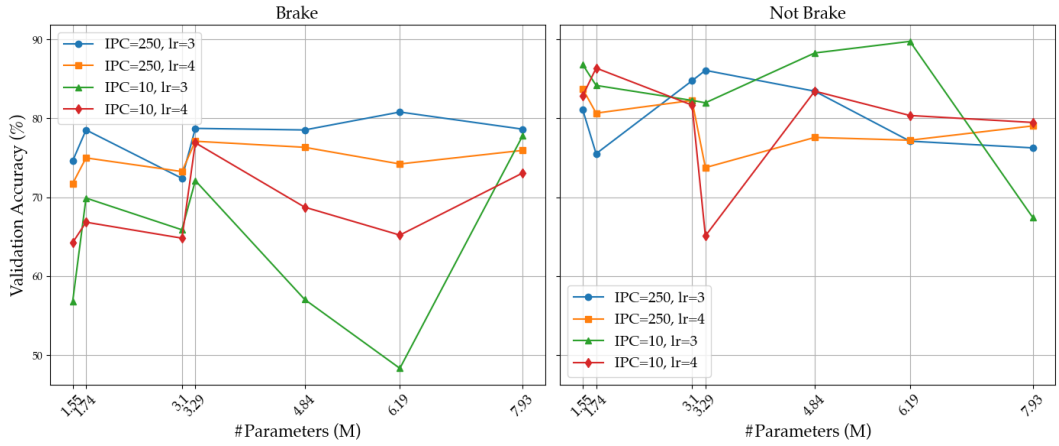


Figure 4.4: Class-specific validation accuracies. Left is for the "brake" command and right is for the "not brake" command.

Compared to other configurations, this setup offers the best balance between the two classes. Therefore, for IPC 10 experiments, we use LoRA rank 8 with DiffFit in further experiments.

Noise Schedule. Another factor influencing the image generation and fine-tuning process is the choice of noise schedule. The previous experiments were based on the classic linear noise schedule. However, since smaller objects in the image play a significant role in driving decisions, we wanted to experiment with the spoon-cos noise schedule, which is better suited for capturing fine details.

As shown in Figure 4.7, the generated examples from both classes using the spoon-cos noise schedule display significant visual differences compared to those generated with a linear noise schedule. The first two rows are visually similar in most cases, while the third row shows sharper details, especially for small objects and textures. The scenes generated with the spoon-cos noise schedule capture more fine-grained elements, such as distant cars and traffic signs, resulting in improved image clarity. However, for low IPC settings, this does not improve validation performance for determining "brake" and "no brake" commands. We believe this is due to the limited number of training samples (20 at IPC 10). Since the spoon-cos noise schedule captures more detailed features, it may lead to overfitting and will not generalize well on the validation dataset.

In Figure 4.7, we show some generated examples from both classes from the experiments using only LoRA, LoRA with DiffFit, and changing the noise schedule to spoon-cos. We can see that the generated images from the first two models look very alike in most cases. In comparison, the images generated with the spoon-cos noise schedule (3rd row) display sharper details, particularly for small objects and textures. The scenes capture more fine-grained elements, like distant cars and signs,

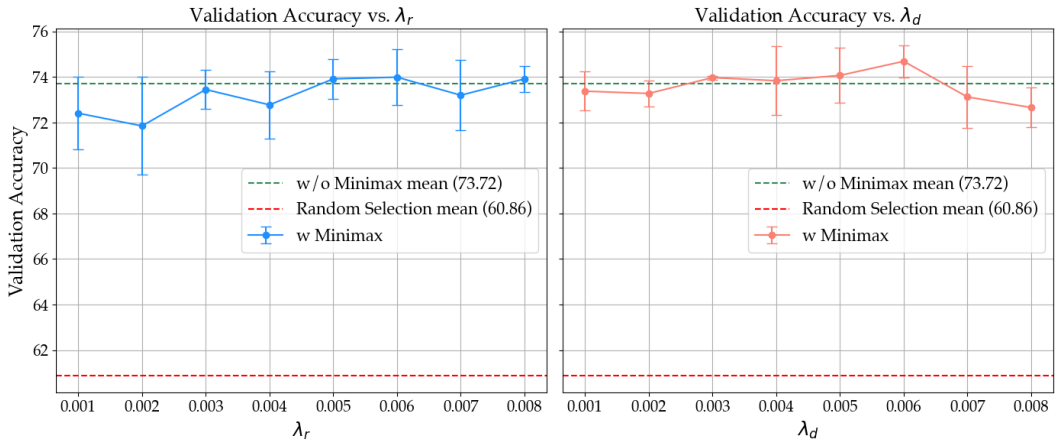


Figure 4.5: Validation accuracy for different weight settings in **IPC 10** experiments. The effects of the weights are explored individually, where λ_r is varied while λ_d is set to zero, and vice versa.

offering better overall image clarity.

4.4 Minimax Fine-tuning on NAVSIM

As confirmed in the minimax paper, random selection often has better performance for higher IPC settings [GVK+24]. So we first explore the effectiveness of the minimax criteria on a lower IPC setting of 10.

4.4.1 Representativeness and Diversity

The minimax objective plays a key role in adapting the diffusion model to the distillation task. In the initial minimax paper, the authors experimented with various IPC settings and weight parameters λ_r, λ_d on the ImageNet dataset, and finally setting $\lambda_r = 0.002, \lambda_d = 0.008$ for IPC 100. Following this setting, we explored the range of 0.001 to 0.008 for IPC 10. As shown in Figure 4.5, the two plots compare the effects of representativeness (λ_r, left) and diversity (λ_d, right) on validation accuracy. Both plots illustrate performance changes across varying parameter values, with each experiment conducted three times. The mean validation accuracies are shown with shaded areas representing standard deviations. Overall, the left plot indicates that tuning λ_r results in greater variability than λ_d. While λ_d yields a slight improvement in validation accuracy, the difference is not substantial. For reference, selecting 10 IPC samples randomly from the original training data achieves an accuracy of approximately 60.86% (with a standard deviation of 6.69%). The large variance in the random selection results may be attributed to the limited number of training samples and the variability in the randomly chosen images. Although

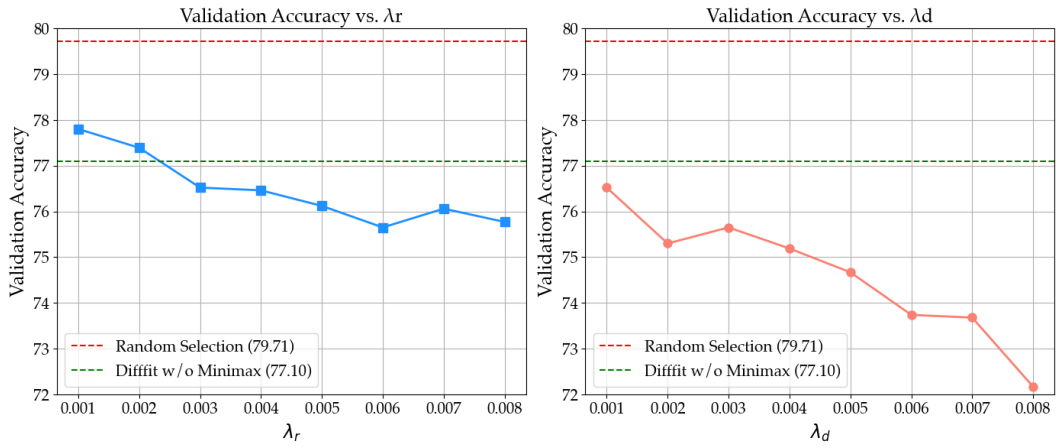


Figure 4.6: Validation accuracy for different weight settings in **IPC 250** experiments.

the training set is evenly sampled, with 10 samples per class, the small dataset size makes it difficult for the model to generalize effectively. Bias may be introduced, as certain samples could be easier or harder to classify, especially for driving datasets where the two classes closely resemble each other. Furthermore, the validation set is not evenly distributed, which amplifies the variability in the validation accuracy. As a result, the model's performance on the validation set fluctuates significantly. The green dashed line marks the mean baseline accuracy of 73.72% (with a standard deviation of 0.87%) achieved without the Minimax criteria, using DiffFit-generated surrogate data. In general, our experiments show that the minimax criteria do not help improve the driving task performance in lower IPC settings.

4.4.2 Different IPC Settings

We now return to the baseline classifier settings, using a batch size 256 to match the configuration used for training on the full dataset. To ensure an appropriate IPC setting, we set the IPC to 250, as the batch size 256 is considerably larger than 10. We see that the random selection already has a good validation accuracy (79.71%) and adding the minimax criteria does not surpass the random line. We consider this to be caused by the large IPC setting of 250. In classical dataset distillation, IPC is often kept no greater than 100. At higher IPC values, random selection is likely sufficient to capture a representative portion of the data distribution, making it challenging for additional refinement through minimax criteria to improve the model's performance.

However, compared to the DiffFit baseline, adding a small degree of representativeness ($\lambda_r = 0.001$) achieves a slight improvement in validation accuracy, increasing it by about 0.7% to reach 77.80%. Our findings suggest that increasing both λ_r and λ_d does not enhance classifier performance. In fact, increasing the diversity weight led

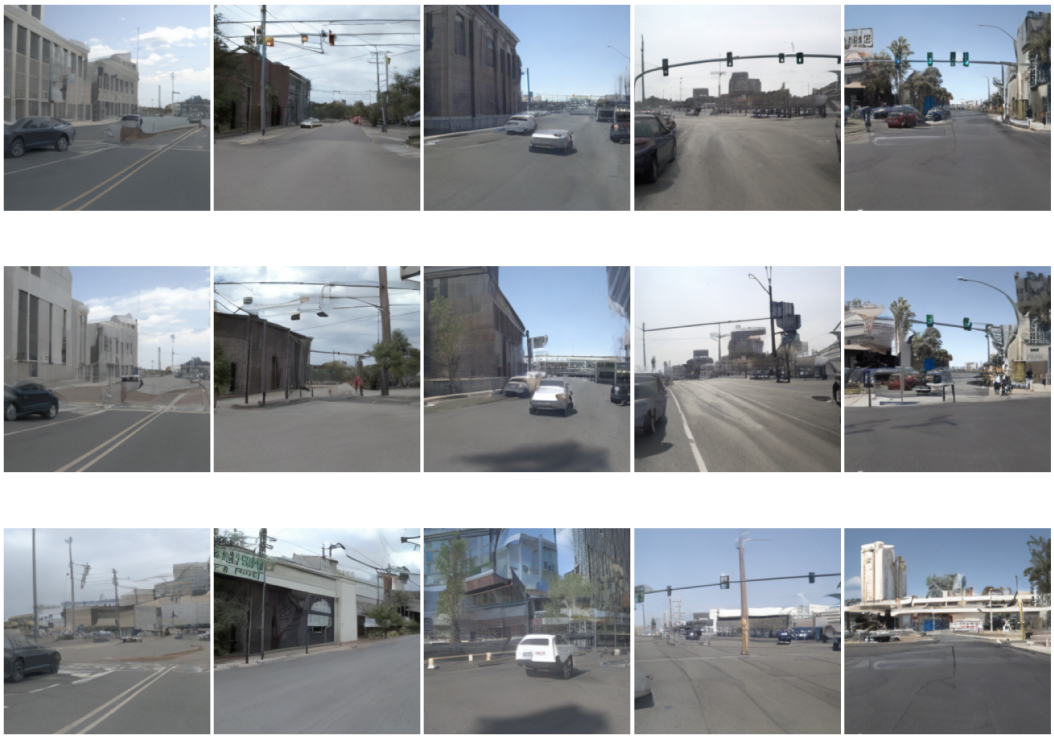
Method	Validation Accuracy	
	IPC 10	**IPC 250**
Random Selection	60.86 ± 6.69	**79.7**
DiffFit	73.72 ± 0.87	77.2
LoRA	71.79 ± 2.31	77.1
DiffFit + LoRA	**74.40 ± 2.35**	77.7
DiffFit + LoRA + Scos	71.81 ± 0.87	78.8

Table 4.3: Comparison of validation accuracies for different IPC settings using different combinations of fine-tuning techniques. Minimax does not help for IPC 10 setting.

to a noticeable decline in performance.

After conducting experiments with various combinations across different IPC settings, we conclude that there is no consistent pattern for achieving the best validation accuracy. Table 4.3 presents an overview of the comparison. For lower IPC settings, the combination of DiffFit and LoRA (rank 8) yields the highest validation accuracy. For higher IPC settings, while it is challenging to surpass the performance of random selection due to its large dataset size, the best results are obtained by combining all the proposed techniques: DiffFit, Minimax, LoRA (rank 32), and the spoon-cosine noise schedule, which yield a validation accuracy of 79.1%. We do not include experiments with Minimax for IPC 10, as it does not improve validation accuracy, even when combined with other fine-tuning settings.

(a)



(b)

Figure 4.7: Some generated **"brake"** images (a) and **"not brake"** images (b) with only LoRA fine-tuned (first row), LoRA and DiffFit fine-tuned (second row) and LoRA and DiffFit fine-tuned with the spoon-cos noise schedule (third row).

5 Conclusion

In this research project, we explore the application of dataset distillation for autonomous driving, an area that has not been deeply studied. We use the state-of-the-art diffusion-based minimax distillation method to generate a distilled dataset that maintains both representativeness and diversity. Our approach adapts the NAVSIM framework, which provides high-resolution images captured from the front-view camera, to integrate with the minimax diffusion distillation pipeline. This allows us to explore the feasibility of dataset distillation in the context of autonomous driving.

During the training process, we experimented with various parameter-efficient fine-tuning techniques, including DiffFit and LoRA, to assess their effectiveness with the minimax criteria. Additionally, we integrate a spoon-cosine noise schedule from the DriveDitFit paper which is tailored to better capture small objects in the complex driving scenes. This noise schedule combines the benefits of a cosine noise schedule at the start of the diffusion process and gradually goes back to a linear noise schedule to preserve the learned distribution in the pre-trained model. It helps the model to retrieve critical fine details such as small vehicles and traffic signs that are relevant for real-world autonomous driving applications. Our results demonstrate that using the combination of these techniques, the performance can be improved.

Although the driving agent in our setup is simplified to perform binary classification (i.e., deciding whether to "brake" or not), it has the potential for full vehicle control if integrated with a proper path-planning module. This means that our distilled dataset and simplified agent have the potential for real-world deployment instead of being an experimental setup. By compressing large, high-dimensional driving datasets into much smaller distilled datasets, our approach sets a baseline for applying dataset distillation in autonomous driving, particularly for cases with limited computational resources or where efficient training is prioritized. We conduct experiments with IPC 10 and IPC 250 settings to evaluate the effectiveness of the distilled datasets across different data sizes. We found that in smaller IPC settings, our distilled approach outperforms both random selection and the DiffFit baseline. However, in larger IPC settings, while our techniques achieved the best results within our framework, they still fell short of matching the performance obtained with random selection from the full dataset. This indicates that while distillation shows promise, there is room for improvement when scaling up to larger datasets.

The observed performance gap between our distilled datasets and a fully trained agent suggests opportunities for further optimization. Since no clear pattern was

identified in tuning the minimax criteria weights to maximize performance, more effective combinations of these parameters may exist. Future work could focus on identifying these optimal weight configurations, which might yield more consistent performance gains. Furthermore, our study focused on a basic binary classification task for the "brake" command, but future research could expand this to include a broader range of driving commands. Expanding to more complex, fine-grained commands would potentially enable the distilled model to make more nuanced decisions that are closer to the decision-making required in real-world autonomous driving scenarios.

From a diffusion perspective, several enhancements could be considered. In our approach, we utilize image-label pairs for distillation and class-conditioned generation. Integrating prompt-based generation could allow for more controlled and finer outputs. As generative models continue to advance, new image generation models are emerging and can also replace the DiT model that we use. For example, the SiT model [MGA+24] based on the DiT architecture has recently become the state-of-the-art model for ImageNet generation. Additionally, reducing the number of timesteps in the diffusion process could also accelerate both training and image generation. With the development of diffusion models capable of single-step generation [YGZ+24], future work might explore whether these newer models could offer even faster and more efficient dataset distillation while maintaining high-quality outputs.

In conclusion, this research project demonstrates that dataset distillation has considerable potential application for autonomous driving. The limitations observed in this research project highlight the need for continued exploration and innovation in distillation techniques, especially as applied to complex, low inter-class variance datasets in safety-critical scenarios like autonomous driving. By refining dataset distillation techniques, future work can help bridge the performance gap and make distilled datasets an efficient data compression for large-scale driving datasets.

Bibliography

[AH23] Fadhel Ayed and Soufiane Hayou. Data pruning and neural scaling laws: fundamental limitations of score-based algorithms, 2023.

[BCNM06] Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Knowledge Discovery and Data Mining*, 2006.

[CBL+19] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019.

[Con23] OpenScene Contributors. Openscene: The largest up-to-date 3d occupancy prediction benchmark in autonomous driving. `https://github.com/OpenDriveLab/OpenScene`, 2023.

[COR+16] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding, 2016.

[CWSH23] Justin Cui, Ruochen Wang, Si Si, and Cho-Jui Hsieh. Scaling up dataset distillation to imagenet-1k with constant memory, 2023.

[CWT+22] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A. Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories, 2022.

[CWT+23] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A. Efros, and Jun-Yan Zhu. Generalizing dataset distillation via deep generative prior, 2023.

[CYM+20] Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning, 2020.

[DHGC23] Daniel Dauner, Marcel Hallgarten, Andreas Geiger, and Kashyap Chitta. Parting with misconceptions about learning-based vehicle motion planning, 2023.

[DHL+24] Daniel Dauner, Marcel Hallgarten, Tianyu Li, Xinshuo Weng, Zhiyu Huang, Zetong Yang, Hongyang Li, Igor Gilitschenski, Boris Ivanovic,

Marco Pavone, Andreas Geiger, and Kashyap Chitta. Navsim: Data-driven non-reactive autonomous vehicle simulation and benchmarking. *arXiv*, 2406.15349, 2024.

[Die00] Thomas G. Dietterich. Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, MCS '00, page 1–15, Berlin, Heidelberg, 2000. Springer-Verlag.

[DN21] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021.

[DR22] Zhiwei Deng and Olga Russakovsky. Remember the past: Distilling datasets into addressable memories for neural networks, 2022.

[Fre99] Robert French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3:128–135, 05 1999.

[GLU12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[GPAM+14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[GVK+24] Jianyang Gu, Saeed Vahidian, Vyacheslav Kungurtsev, Haonan Wang, Wei Jiang, Yang You, and Yiran Chen. Efficient dataset distillation via minimax diffusion, 2024.

[GZB22] Chengcheng Guo, Bo Zhao, and Yanbing Bai. Deepcore: A comprehensive library for coreset selection in deep learning, 2022.

[HC21] K. Tan et al. H. Caesar, J. Kabzan. Nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. In *CVPR ADP3 workshop*, 2021.

[HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *CoRR*, abs/2006.11239, 2020.

[HS22] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022.

[HSW+21] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

[HVD15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[KAAL22] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models, 2022.

[KKO+22] Jang-Hyun Kim, Jinuk Kim, Seong Joon Oh, Sangdoo Yun, Hwanjun Song, Joonhyun Jeong, Jung-Woo Ha, and Hyun Oh Song. Dataset condensation via efficient synthetic-data parameterization, 2022.

[Kri09] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[KSW15] Diederik P. Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick, 2015.

[KW22] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

[LHAR22] Noel Loo, Ramin Hasani, Alexander Amini, and Daniela Rus. Efficient dataset distillation using random feature approximation, 2022.

[MBL20] Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models, 2020.

[MGA+24] Nanye Ma, Mark Goldstein, Michael S. Albergo, Nicholas M. Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers, 2024.

[NCL21] Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. Dataset meta-learning from kernel ridge-regression, 2021.

[PX23] William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023.

[STW+23] Chonghao Sima, Wenwen Tong, Tai Wang, Li Chen, Silei Wu, Hanming Deng, Yi Gu, Lewei Lu, Ping Luo, Dahua Lin, and Hongyang Li. Scene as occupancy. 2023.

[TJZ+24] Jiahang Tu, Wei Ji, Hanbin Zhao, Chao Zhang, Roger Zimmermann, and Hui Qian. Driveditfit: Fine-tuning diffusion transformers for autonomous driving. *arXiv preprint arXiv:2407.15661*, 2024.

[TSdC+19] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. An empirical study of example forgetting during deep neural network learning, 2019.

[VSP+23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[Wel09] Max Welling. Herding dynamical weights to learn. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 1121–1128, New York, NY, USA, 2009. Association for Computing Machinery.

[WZTE18] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018.

[XYS+23] Enze Xie, Lewei Yao, Han Shi, Zhili Liu, Daquan Zhou, Zhaoqiang Liu, Jiawei Li, and Zhenguo Li. Difffit: Unlocking transferability of large diffusion models via simple parameter-efficient fine-tuning, 2023.

[YCSL23] Zetong Yang, Li Chen, Yanan Sun, and Hongyang Li. Visual point cloud forecasting enables scalable autonomous driving. *arXiv preprint arXiv:2312.17655*, 2023.

[YGZ+24] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T. Freeman, and Taesung Park. One-step diffusion with distribution matching distillation, 2024.

[ZB21] Bo Zhao and Hakan Bilen. Dataset condensation with differentiable siamese augmentation, 2021.

[ZB22a] Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching, 2022.

[ZB22b] Bo Zhao and Hakan Bilen. Synthesizing informative training samples with gan, 2022.

[ZMB21] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching, 2021.

[ZNB22] Yongchao Zhou, Ehsan Nezhadarya, and Jimmy Ba. Dataset distillation using neural feature regression, 2022.