# UNDERSTANDING AND IMPROVING THE TRAINING DYNAMICS OF EDM2

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Diffusion models have achieved remarkable success in image generation, yet their training dynamics can strongly affect both convergence speed and the persistence of high-quality generation. In this work, we investigate the optimization dynamics of EDM2 on small-scale image generation benchmarks and identify factors that influence its generation quality and training stability. We introduce several improvements, including learning-rate scheduling, activation statistics regulation, adaptive reconstruction objectives, and modifications to EDM2-specific components, which improve generation quality and extend the persistence of high-performance states. Furthermore, we analyze the optimization geometry induced by the EDM2 parameterization and find that, compared with an EDM-style control model, it exhibits lower sensitivity to relative parameter perturbations during early training. These results highlight the importance of considering optimization dynamics beyond final performance when improving diffusion models.

## 1 INTRODUCTION

Diffusion models (Ho et al., 2020; Nichol & Dhariwal, 2021; Sohl-Dickstein et al., 2015; Song et al., 2021a; Song & Ermon, 2019; Song et al., 2021b; Vincent, 2011; Lipman et al., 2023; Liu et al., 2023; Geng et al., 2025) have recently achieved remarkable progress in high-quality image generation and have been extended to various modalities, including text (Nie et al., 2025), audio (Kong et al., 2021; Popov et al., 2021), video (Kong et al., 2025; Guo et al., 2024; Brooks et al., 2024; Chen et al., 2025; Ho et al., 2022; Yin et al., 2025), and 3D content (Tang et al., 2024; Lin et al., 2023; Poole et al., 2023; Raj et al., 2023; Shue et al., 2023). Despite these advances, efficiently optimizing diffusion models remains challenging due to the stochastic nature of noise perturbations, multi-scale denoising processes, and complex optimization dynamics.

Recent studies have shown that architectural and normalization designs can substantially improve diffusion model training and generation quality. EDM2 (Karras et al., 2024) exemplifies this direction by redesigning the network architecture with weight normalization (Salimans & Kingma, 2016), pixel normalization (Karras et al., 2018), and magnitude-preserving (MP) operations. These components effectively control activation magnitudes and enable more efficient optimization, allowing EDM2 to achieve strong performance on large-scale image generation benchmarks. However, existing EDM2 studies mainly focus on signal propagation and scalability, while its training dynamics and long-term optimization behavior remain less explored.

In this work, we first investigate the training dynamics of EDM2 and explore improvements from multiple perspectives while preserving its parameterization principles. Specifically, we study data augmentation, learning-rate scheduling, activation statistics, and reconstruction objectives to improve generation quality across CIFAR-10 (Krizhevsky et al., 2009), AFHQv2-64 (Choi et al., 2020), and FFHQ-64 (Karras et al., 2019). Through these experiments, we observe that rapid convergence to strong generation quality does not necessarily lead to persistent high performance: as training continues, the FID of EDM2 can degrade or fluctuate after reaching an early optimum, particularly on CIFAR-10 and AFHQv2-64, while this behavior is less pronounced on FFHQ-64. Figure 1 provides an overview of these training trajectories and the effects of our interventions across the three benchmarks. We then investigate the factors associated with the limited persistence of high-performance states and introduce additional stability-oriented modifications to EDM2. All of our



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

improvements, together with a comparison with the EDM (Karras et al., 2022), are summarized in Table 1.

We then examine the optimization dynamics underlying the observed training behavior. In particular, we analyze the local optimization geometry induced by the EDM2 parameterization and compare it with an EDM-style control model. Since the two models operate at fundamentally different parameter scales, directly comparing raw Hessian curvature can be misleading. We therefore introduce a scale-corrected optimization sensitivity measure by combining Hessian curvature with parameter magnitude. Our analysis shows that EDM2 exhibits lower sensitivity to relative parameter perturbations during early-to-middle stage of training. This reduced sensitivity is consistent with EDM2's ability to use larger learning rates in early-to-middle stage of training, allowing larger optimization steps and thereby contributing to faster convergence.

Table 1: Evaluation of our training improvements and comparison with the EDM baseline. Lower FID is better; NFE denotes the number of function evaluations. The values without parentheses are the FID scores obtained using the fixed random seed of the main experiments. For Configurations C–E, we additionally evaluate the same selected checkpoints using four additional sampling seeds and report the mean and standard deviation over all five sampling seeds in parentheses.

| **Training configuration** | **CIFAR-10** (Cond.) | **AFHQv2** (Uncond.) | **FFHQ** (Uncond.) |
|---|---|---|---|
| A Baseline (EDM2) | 2.05 | 2.82 | 3.08 |
| B +Non-leaky Data Augmentation | 1.85 | 2.46 | 2.90 |
| C +Learning Rate Schedule | 1.72 (1.73 $\pm$ 0.01) | 2.01 (2.01 $\pm$ 0.02) | 2.04 (2.04 $\pm$ 0.02) |
| D +Weight Centralization | 1.70 (1.72 $\pm$ 0.02) | 1.98 (1.98 $\pm$ 0.01) | 2.02 (2.00 $\pm$ 0.03) |
| E +Adaptive Frequency-aware Loss | 1.70 (1.73 $\pm$ 0.03) | **1.93** (1.91 $\pm$ 0.02) | **1.99** (1.98 $\pm$ 0.03) |
| F +Late-stage stability | **1.66** | 2.00 | – |
| EDM | 1.79 | 1.96 | 2.39 |
| NFE | 35 | 79 | 79 |

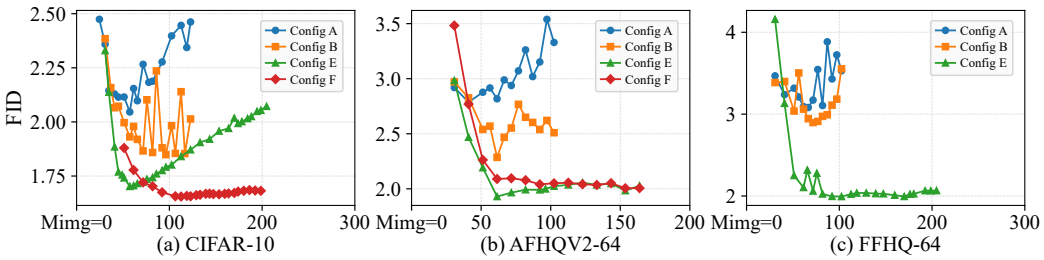



Figure 1: FID trajectories of Configurations A, B, and E, together with Configuration F incorporating the stability-oriented interventions described in Sec. 2.5, on CIFAR-10, AFHQv2-64, and FFHQ-64. Configurations C and D are shown together for comparison in Appendix C for clarity, as their overall trajectories are similar to Configuration E, although their absolute FID values differ.

## 2 Improving EDM2 Training Dynamics

This section investigates how to improve the training stability and generation quality of EDM2 on small-scale image generation benchmarks. We use the EDM2 network implemented within the EDM framework as our baseline. Unlike conventional U-Net (Ronneberger et al., 2015) architectures that commonly employ Group Normalization, EDM2 replaces it with weight normalization, pixel normalization, and magnitude-preserving (MP) operations, maintaining stable activation statistics while mitigating the uncontrolled growth of activation magnitudes during training. EDM2 also introduces several additional architectural modifications, summarized in Appendix B.1. We progressively construct our improved training scheme through multiple stages, with implementation details, hyperparameter settings, and mathematical derivations provided in Appendix B.

### 2.1 BASELINE (CONFIGURATION A)

Since EDM2 does not provide official configurations for CIFAR-10, FFHQ-64, or AFHQv2-64, we construct our baseline by combining the EDM2 network architecture with the training scheme adopted in EDM. The resulting baseline achieves FID scores (Heusel et al., 2017) of 2.05, 2.82, and 3.08 on CIFAR-10, AFHQv2-64, and FFHQ-64, respectively.

### 2.2 INITIAL IMPROVEMENTS

Based on the baseline configuration, we first investigate several simple strategies to improve the training performance of EDM2 on small-scale image generation benchmarks. These modifications preserve the original EDM2 architecture and mainly focus on improving data utilization and optimization dynamics.

#### 2.2.1 NON-LEAKY DATA AUGMENTATION (CONFIGURATION B)

Diffusion models trained on small-scale datasets often suffer from limited data diversity and degraded generalization. To alleviate this issue, we adopt the non-leaky data augmentation pipeline proposed by Karras et al. (2020). The augmented image is provided as the network input together with its augmentation parameters as additional conditioning, preventing the model from treating augmentation-induced variations as intrinsic characteristics of the data distribution. The augmentation conditioning pathway is further modified to follow the magnitude-preserving design principle of EDM2, with implementation details provided in Appendix B.2. With this modification, the FID is reduced to 1.85, 2.46, and 2.90 on CIFAR-10, AFHQv2-64, and FFHQ-64, respectively.

#### 2.2.2 LEARNING RATE SCHEDULE IMPROVEMENT (CONFIGURATION C)

EDM2 employs forced weight normalization to decouple the effective learning rate from weight magnitudes. It further adopts an inverse-square-root decay schedule (Kingma & Ba, 2015), $\alpha(t) = \alpha_{\text{ref}}/\sqrt{\max(t/t_{\text{ref}}, 1)}$ which actively reduces the learning rate during training. We observe that the EMA length corresponding to the best FID remains relatively short, indicating limited parameter smoothing by EMA and suggesting that the learning rate may remain relatively high during late-stage optimization.

To address this issue, we introduce an explicit learning-rate annealing mechanism in the later training stage. We compare cosine and piecewise annealing and adopt piecewise annealing because it provides direct control over the decay trajectory at different training stages. The original EDM2 schedule is retained during early training, and annealing is enabled only after the model enters the later optimization stage. Detailed settings are provided in Appendix B.3. With this modification, the FID is reduced to 1.72, 2.01, and 2.04 on CIFAR-10, AFHQv2-64, and FFHQ-64, respectively.

### 2.3 ACTIVATION MEAN OFFSET AND WEIGHT CENTRALIZATION

#### 2.3.1 MOTIVATION: ACTIVATION MEAN OFFSET IN EDM2

Zhang & Sennrich (2019) showed that re-scaling invariance plays a more critical role than re-centering invariance. The design of EDM2 is consistent with this conclusion. Karras et al. (2024) found from training dynamics that activation magnitudes vary substantially more than activation means, highlighting the importance of controlling activation magnitudes throughout training. Consequently, EDM2 focuses on maintaining stable activation magnitudes through weight normalization, pixel normalization, and magnitude-preserving (MP) operations. In this work, activation magnitude refers to the RMS of intermediate activations (see Appendix A.5).

However, from the perspective of signal propagation, the identity $\text{RMS}^2 = \mu^2 + \sigma^2$ implies that, when magnitude-preserving operations such as pixel normalization maintain the activation RMS at a relatively stable scale, a nonzero activation mean may reduce the variance of the intermediate features. Furthermore, unlike the machine translation and other tasks studied by Zhang & Sennrich (2019), EDM-based models operate on zero-mean normalized image signals: both the noisy input and denoising target are zero-mean in expectation. Therefore, we conjecture that intermediate features may not be desirable to deviate excessively from zero mean.

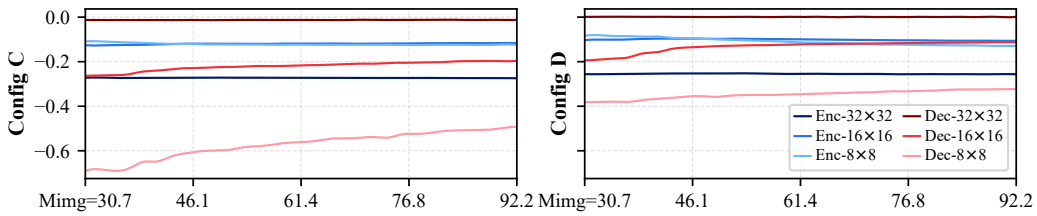



Figure 2: Activation mean trajectories at different network depths over training for Configurations C and D on CIFAR-10. Each curve shows the evolution of the activation mean at a specific network depth throughout training. Both panels use identical y-axis limits and tick scales to enable direct comparison between the two configurations. Results for FFHQ-64 and AFHQv2-64 are provided in Appendix B.4.

Although EDM2 does not explicitly constrain activation means, its design does not imply that activation means continuously drift away from zero. To empirically investigate this phenomenon, we measure activation means at different network depths during training under Configuration C on CIFAR-10. As shown in Figure 2, most blocks maintain activation means close to zero throughout the measured training period, with no increasing trend in the offsets. However, Enc-$32 \times 32$ and Dec-$8 \times 8$ retain relatively large activation mean offsets, with the latter showing slightly larger variation during training.

Based on these observations, we introduce a lightweight centralization mechanism to reduce remaining activation mean offsets while preserving EDM2's magnitude-preserving design. We do not directly impose explicit centering on activations, as such constraints may be overly restrictive. Instead, following the design philosophy of EDM2, we perform centralization at the weight level to reduce the expected mean shift of pre-activation signals while preserving the original magnitude-preserving formulation. The detailed implementation is described in Sec. 2.3.2. We further evaluate whether weight centralization can reduce the remaining activation mean offsets. As shown in Figure 2, the decoder blocks exhibit a clearer reduction in mean offset after applying weight centralization. Moreover, the variation of these mean offsets during training is reduced.

### 2.3.2 Weight Centralization for Activation Stabilization (Configuration D)

We first center the weight vector $\mathbf{w}$ prior to weight normalization:

$$\tilde{\mathbf{w}} = \mathbf{w} - \mathrm{mean}(\mathbf{w}), \tag{1}$$

where $\mathrm{mean}(\cdot)$ computes the scalar mean over all elements of $\mathbf{w}$. This operation removes the component of each convolutional weight vector along the all-ones direction. Since MP-SiLU introduces an asymmetric transformation that shifts activation means away from zero, we apply weight centralization to the convolutional layers following MP-SiLU activations within residual branches, while leaving the remaining layers unchanged. These layers account for more than $88\%$ of the trainable parameters in our models and therefore cover the majority of parameters directly affected by this nonlinear transformation. The remaining parameters mainly belong to embedding layers and layers without preceding MP-SiLU activations. The inputs and outputs of embedding layers do not share the same zero-centered signal parameterization as image features, while layers without preceding MP-SiLU activations are less directly associated with this source of mean offsets. Therefore, we do not apply weight centralization to these components.

Since EDM2 employs forced weight normalization to prevent the effective learning rate from decreasing during training, we retain this mechanism after introducing weight centralization. To preserve this property under the modified parameterization, the weight vector is first centered and then normalized:

$$\hat{\mathbf{w}}_i = \sqrt{d}\frac{\tilde{\mathbf{w}}_i}{\|\tilde{\mathbf{w}}_i\|_2}. \tag{2}$$

A detailed analysis of this modification and the complete gradient analysis of the combined weight centralization and normalization operations are provided in Appendix B.4. Weight centralization provides an additional improvement in generation quality, achieving FID scores of 1.70 / 1.98 / 2.02 on CIFAR-10, AFHQv2-64, and FFHQ-64, respectively.

### 2.4 Adaptive Frequency-aware Reconstruction Objective (Configuration E)

The standard EDM framework optimizes a pixel-wise mean squared error (MSE) objective, which assigns identical weights to all dimensions of the predicted and target images. Although this uniform weighting provides a simple and effective reconstruction objective, we hypothesize that it may not be optimal for image generation. Therefore, we introduce a structured frequency-aware weighting scheme into the reconstruction objective. The frequency domain provides a natural parameterization for such weighting. It is well known that neural networks generally follow the frequency principle: low-frequency structures are typically learned earlier during optimization, whereas high-frequency details are acquired later and are more difficult to optimize. This suggests that the relative importance of different frequency components may need to evolve throughout training. Moreover, previous studies (Kim et al., 2025; Kadkhodaie et al., 2024; Zhang & Hooi, 2023; Yang et al., 2023) have shown that high-frequency features are important for modeling diffusion-based image datasets.

However, existing approaches primarily focus on high-frequency enhancement and do not directly investigate frequency weighting for diffusion models trained from scratch across different noise levels and training stages. Since low-frequency structures are generally learned well by neural networks, we focus on adaptively weighting middle-frequency components rather than revisiting high-frequency enhancement. Our formulation can also accommodate high-frequency emphasis at later training stages by adjusting the frequency range. Therefore, inspired by Kim et al. (2025), we introduce a frequency-transformed reconstruction objective into the EDM training framework. Specifically, we define

$$\mathcal{L} = \|\phi\left(D_\theta(x_\sigma, \sigma)\right) - \phi(y)\|_2^2, \tag{3}$$

where $D_\theta$ denotes the denoising network, $x_\sigma$ denotes the noisy input at noise level $\sigma$, and $y$ represents the corresponding clean image. The transformation operator is defined as $\phi = I + \lambda\mathrm{BPF}$ where BPF denotes a band-pass filtering operator, with implementation details provided in Appendix B.5.2. Unlike formulations that directly apply high-pass transformations, our design treats the frequency band emphasized by the band-pass filter as a tunable hyperparameter. This allows the model to selectively enhance task-relevant frequency ranges during optimization. Moreover, when $\phi$ is an invertible linear transformation, the optimal denoising solution in the output space is preserved, while the reconstruction metric is modified. The corresponding analysis is provided in Appendix B.5.1.

**Noise-dependent weighting** A fixed frequency transformation may not be optimal across all diffusion noise levels. Inspired by the noise-dependent loss weighting used in EDM, we further modulate the strength of the frequency-aware component according to the noise level. At high noise levels, the input contains insufficient information to reliably reconstruct fine structures. Excessively emphasizing frequency components in this regime may introduce noisy and unreliable optimization signals. Conversely, at very low noise levels, the model is required to recover increasingly detailed structures, and overly strong frequency constraints may lead to unstable high-frequency artifacts. To address these effects, we introduce a noise-dependent weighting factor:

$$\lambda_\sigma = \exp\left(-\frac{(\log\sigma - P_{\mathrm{mean}})^2}{2P_{\mathrm{std}}^2}\right). \tag{4}$$

This weighting increases the contribution of the frequency-aware component around the central noise range specified by $P_{\mathrm{mean}}$, while gradually suppressing its contribution at extremely high and low noise levels. Importantly, $\lambda_\sigma$ only adjusts the additional frequency-aware component and does not replace the original noise-dependent weighting used by EDM.

**Training-progress-dependent weighting** The appropriate frequency weighting may also change throughout optimization. During early training, neural networks primarily learn coarse image structures, and introducing strong frequency constraints too early may interfere with this process. Moreover, different frequency ranges may require different emphasis at different optimization stages. We therefore introduce a training-progress-dependent factor:

$$\lambda_t = \frac{\log(1+t)}{\log(1+T_{\max})}, \tag{5}$$

where $t$ denotes the number of training images processed so far. This logarithmic schedule gradually increases the contribution of the frequency-aware component during training, allowing the model to

Table 2: Ablation study of the proposed frequency-aware objective and its scheduling strategies. FID values are reported for each configuration, with lower values indicating better performance.

| Training configuration | CIFAR-10 | AFHQv2-64 | FFHQ-64 |
|---|---|---|---|
| Fixed frequency-aware objective | 1.71 | 2.00 | 2.04 |
| + Noise-dependent weighting | 1.71 | 1.97 | 2.19 |
| + Training-progress-dependent weighting (Full) | **1.70** | **1.93** | **1.99** |

first establish coarse structures and subsequently emphasize middle-frequency information. The parameter $T_{\max}$ controls the increasing rate of the frequency-aware contribution. By adjusting the frequency range of the band-pass filter (BPF) and the corresponding $\lambda_t$ schedule, the formulation can also accommodate stronger emphasis on high-frequency components during later training stages.

Combining the overall frequency strength, noise-dependent weighting, and training-progress-dependent weighting, the final transformation becomes: $\phi = I + \lambda\lambda_t\lambda_\sigma \mathrm{BPF}$. Here, $\lambda$, $\lambda_t$, and $\lambda_\sigma$ control the overall strength of the frequency-aware component, its dependence on training progress, and its dependence on diffusion noise level, respectively.

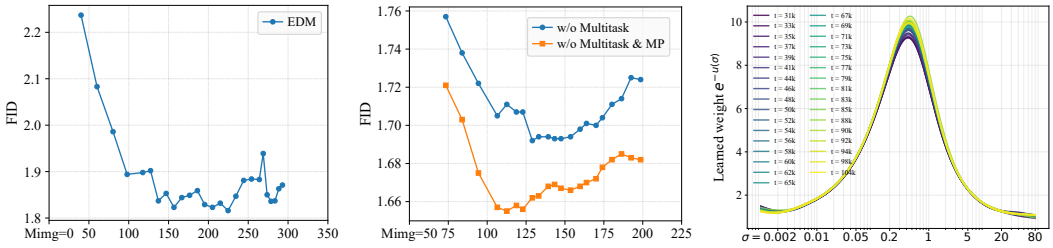
**Ablation** As shown in Table 2, the effectiveness of the proposed objective varies across datasets. On CIFAR-10, it provides limited improvement. On FFHQ-64, applying only the noise-dependent weighting degrades the FID from 2.04 to 2.19, suggesting that this weighting schedule may induce a less favorable optimization trajectory for this dataset. Incorporating the training-progress weighting $\lambda_t$ alleviates this degradation. Hyperparameter settings and search ranges are provided in Appendix B.5.3. The final FID is 1.70, 1.93, and 1.99 on CIFAR-10, AFHQv2-64, and FFHQ-64, respectively.

### 2.5 Late-Stage Stability of EDM2

The modifications above lead to Config. E, which substantially improves FID over the EDM2 baseline. However, the FID trajectories reveal a late-stage degradation or fluctuation that cannot be captured by the minimum FID alone. Figure 1 visualizes the configurations with more distinct FID trajectories, including Configurations A, B, and E, as well as Config. F with the stability-oriented interventions described below. We first investigate this phenomenon on CIFAR-10, where it is most pronounced, and then evaluate whether the resulting intervention transfers to other datasets. Figure 3 further compares the late-stage FID trajectories of EDM2 and the original EDM with the SongUNet architecture on CIFAR-10.

This comparison indicates that late-stage FID variation may be a general characteristic of continued diffusion-model optimization rather than a phenomenon introduced exclusively by EDM2. The distinctive behavior of EDM2 in our experiments is the combination of rapid convergence and a relatively short-lived high-performance regime. Consequently, the observed performance becomes more sensitive to checkpoint selection. This motivates us to distinguish two aspects of training dynamics: the speed at which a model reaches a high-performance regime and the persistence of that regime after reaching it. The former is an important advantage of EDM2 that we aim to preserve, whereas the latter motivates the following stability analysis.

We initially considered three possible explanations for this behavior. First, activation mean offsets may contribute to late-stage instability. However, the FID trajectories of Config. C and Config. D remain highly similar, suggesting that weight centralization does not substantially alter this behavior. Therefore, activation mean offset is unlikely to be the dominant factor. Second, EDM2 employs a substantially larger learning rate than EDM, which may contribute to its faster convergence. However, this explanation alone cannot account for the shorter persistence of the high-performance regime: when Config. C enters this regime, its learning rate has already been substantially reduced through piecewise annealing, yet its high-performance regime remains shorter than that observed for EDM. Third, continued optimization may lead to overfitting, particularly in the low-noise regime. However, this explanation is also insufficient to fully explain the behavior, since the EDM framework with the SongUNet architecture and data augmentation can maintain a relatively long high-performance regime under similar training conditions. These observations motivate us to investigate




(a) (b) (c)

Figure 3: FID trajectories and loss weighting dynamics. (a) FID trajectory of the original EDM with the SongUNet architecture over 300M training images on CIFAR-10. (b) Late-stage FID trajectories on CIFAR-10 for progressive ablations of EDM2 components. (c) Loss weighting $e^{-u(\sigma)}$ at different training stages across $\sigma \in [0, 80]$.

whether specific components introduced by EDM2 affect the persistence of high-performance solutions.

**Progressive ablation of EDM2 components** Starting from Config. E, we progressively remove components introduced by EDM2 and retain a modification only when its removal substantially improves the persistence of high performance. This procedure allows us to focus on components that affect the temporal behavior of training rather than reporting negative ablations without clear effects.

The first component that produces a clear effect is the uncertainty-based multi-task learning objective. Compared with EDM, which uses fixed noise-dependent weighting, EDM2 introduces learnable uncertainty-based weights that adaptively adjust the relative contribution of different noise levels during training (see Appendix B.1.6 for details).

We consider two possible mechanisms through which this adaptive weighting may affect optimization dynamics. First, the learned weights may change substantially during training, continuously shifting the relative optimization emphasis across noise levels. Second, the weighting may exceed one for certain noise levels, thereby amplifying their gradient contributions and effectively increasing the optimization step size for those samples. Therefore, we analyze the additional noise-dependent weighting term introduced by uncertainty-based weighting in EDM2, $e^{-u(\sigma)}$. As shown in Figure 3c, the weighting curves remain highly similar across training stages, providing little evidence that this weighting undergoes substantial temporal drift. Therefore, our measurements do not support the first hypothesis. Moreover, $e^{-u(\sigma)}$ becomes substantially larger than one in the low-noise regime. Around $\sigma \approx 0.5$, the weighting reaches approximately 9. Since this factor directly scales the reconstruction loss and its corresponding gradients, it effectively increases the optimization step size of samples in this noise regime. This may contribute to EDM2's rapid convergence while also causing the model to traverse the high-performance regime more quickly, thereby reducing its persistence. Removing uncertainty-based weighting substantially changes the FID trajectory. As shown in Figure 3b, the model reaches its best FID later after removing this weighting, but the resulting high-performance regime becomes substantially longer.

A second observation concerns the MP components in EDM2, whose specific implementations are described in Appendix B.1.5. After modifying these components, the resulting FID trajectories are shown in Figure 3b. The modified model reaches its best FID at a later stage of training, but its high-performance regime is substantially extended compared with the original configuration. The specific modifications are described in Appendix B.6. While this result indicates that the MP components are closely related to the temporal evolution of generation quality, the underlying reason why they lead to a relatively short high-performance regime remains unclear.

The remaining EDM2 components either have limited effects on the late-stage trajectory or increase FID. We therefore omit these negative ablations from the main text. The resulting stability-oriented configuration is denoted as Config. F. Notably, Config. F is not a universally superior replacement for Config. E. Instead, it sacrifices part of EDM2's rapid convergence in exchange for improved

temporal stability. We further conduct three robustness checks to exclude alternative explanations for the observed late-stage degradation; details are provided in Appendix B.7.

**Dataset dependence** The severity of late-stage instability varies substantially across datasets. On CIFAR-10 and AFHQv2-64, late-stage fluctuations or degradation are more apparent, whereas the FID trajectory of Config. E on FFHQ-64 remains considerably smoother over the observed training range. Therefore, additional stability intervention is unnecessary for this dataset. We further apply the stability-oriented modifications of Config. F to AFHQv2-64. As shown in Figure 1, Config. F produces a smoother trajectory and maintains performance more consistently, but at the cost of a slightly higher FID than Config. E. This result demonstrates that improving temporal stability does not necessarily lead to improved minimum FID. We therefore view Config. F as a targeted intervention for training settings where the high-performance regime is unusually short, rather than as a universal replacement for Config. E.

## 3 Analyzing Loss Landscape and Optimization Stability

Karras et al. (2024) adopt explicit weight normalization (WN), magnitude-preserving (MP) operators, and lightweight pixel normalization in the encoder, effectively replacing conventional group normalization. The original motivation of this design is primarily from the perspective of signal propagation, aiming to maintain controlled signal magnitudes across network layers during training.

However, existing optimization theory (Santurkar et al., 2018; Dukler et al., 2020) provides another perspective: normalization techniques can affect optimization not only through forward signal statistics, but also through the induced parameterization and local curvature. Motivated by this observation, we investigate whether the architectural redesign of EDM2 also changes its local optimization geometry. Specifically, we measure the leading five eigenvalues of the Hessian in a shared backbone parameter subspace of the EDM2-style and EDM-style networks. This subspace contains more than 88% of the trainable parameters and consists of functionally matched convolutional layers in the residual paths, allowing a controlled comparison while avoiding parameters with substantially different roles and scales. Detailed experimental settings and the motivation for this subspace selection are provided in Appendix C.1.

Since the two networks employ different parameterizations and therefore operate at different parameter scales, directly comparing Hessian eigenvalues can be misleading. We therefore account for parameter scale by considering the maximum local gradient change induced by a fixed relative parameter perturbation. For a perturbation satisfying $\|\Delta\theta\|_2 = \epsilon\|\theta\|_2$, the local gradient change is approximately $H\Delta\theta$, and its maximum magnitude scales as $\epsilon\lambda_{\max}(H)\|\theta\|_2$ in the leading-curvature direction. We thus define the scale-corrected local sensitivity as

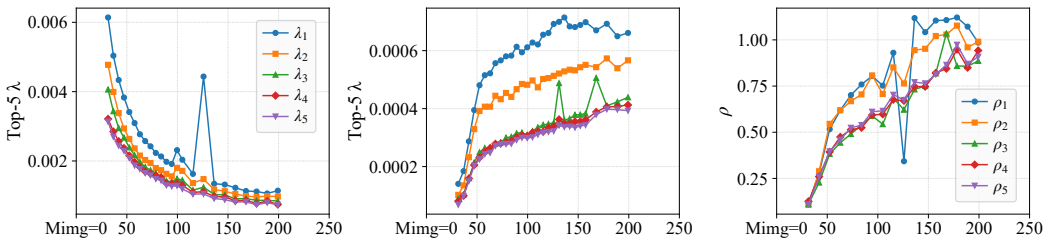
$$\mathcal{S}(H,\theta) \triangleq \lambda_{\max}(H)\|\theta\|_2, \tag{6}$$

where $H$ denotes the Hessian restricted to the selected parameter subspace and $\theta$ denotes the corresponding parameter vector. We then use the ratio

$$\rho = \frac{\mathcal{S}(H_{\mathrm{EDM2}}, \theta_{\mathrm{EDM2}})}{\mathcal{S}(H_{\mathrm{EDM}}, \theta_{\mathrm{EDM}})}, \tag{7}$$

to compare the relative local sensitivity of the two parameterizations. Thus, $\rho < 1$ indicates that EDM2 exhibits smaller local gradient changes under the same relative parameter perturbation. Detailed derivations are provided in Appendix C.2.

The results are shown in Figure 4. Under the same relative parameter perturbation, EDM2 exhibits substantially lower local gradient sensitivity during the early-to-middle stage of training. As training proceeds, the difference between the two networks gradually decreases, and the ratio exceeds one in the later stage, when EDM2 has already entered the FID degradation stage. Comparisons of the other leading Hessian eigenvalues are provided as additional reference. These results indicate that the optimization advantage of EDM2 is most pronounced during the early-to-middle stage of training, but gradually diminishes as training progresses. This suggests that the EDM2 architecture may




(a) EDM (b) EDM2 (c) Scale-corrected sensitivity ratios $\rho_i$

Figure 4: Hessian-based analysis of local optimization geometry on CIFAR-10. (a) Top-5 eigenvalues of the Hessian in the shared backbone parameter subspace of the EDM-style control network from 30M to 200M training images. (b) Corresponding top-5 Hessian eigenvalues of the EDM2-style network in the same parameter subspace and training range. (c) Scale-corrected sensitivity ratios $\rho_i$ comparing the corresponding $i$-th Hessian eigenvalues of EDM2 and EDM. Specifically, $\rho_1$ is computed from the largest Hessian eigenvalues $\lambda_1 = \lambda_{\max}$ of the two networks, The latter ratios are included for reference.

allow larger learning rates and parameter updates during the early-to-middle stage without causing comparably large local changes in the gradient, whereas the learning rate should be gradually reduced during later training.

## 4 Conclusion

In this work, we investigate the training and optimization dynamics of EDM2 on small-scale image generation benchmarks. Although EDM2 introduces effective architectural designs that substantially improve signal propagation and accelerate convergence, we systematically explore several directions to further improve its training and generation performance, including learning-rate scheduling, data augmentation, activation statistics, and adaptive objective design. These modifications improve generation quality across multiple benchmarks while preserving the overall EDM2 parameterization principles. Through these experiments, we observe that rapid convergence to strong generation quality does not necessarily lead to persistent high performance, as the high-performance regime can be relatively short-lived in certain settings. We therefore further identify factors associated with late-stage performance degradation and introduce stability-oriented modifications that extend the high-performance regime.

Beyond these empirical improvements, we analyze the optimization geometry induced by the EDM2 parameterization. By comparing EDM2 with an EDM-style control model under scale-corrected parameter perturbations, we show that EDM2 exhibits lower scale-corrected optimization sensitivity during early training. This provides a new perspective for understanding its faster convergence behavior.

Our experimental comparisons are intentionally centered on EDM2 and its closely matched configurations. EDM2 has already been extensively evaluated against representative diffusion models in its original work, whereas our goal is not to establish a new state-of-the-art benchmark, but to further investigate and improve the training and optimization dynamics of EDM2.

Nevertheless, our study has several limitations. First, our experiments are conducted on relatively small-scale image generation benchmarks, including CIFAR-10, FFHQ-64, and AFHQv2-64. Although these benchmarks allow controlled investigation of training dynamics and optimization behavior, whether these training dynamics and the effectiveness of our proposed modifications persist at larger image resolutions and dataset scales remains to be established. Second, our study focuses on the EDM2 architecture and its associated parameterization. It remains an open question whether our proposed improvements remain effective for other diffusion model architectures, such as DiT (Peebles & Xie, 2023), and for other diffusion formulations, such as flow matching (Lipman et al., 2023). Extending our study to these settings would help assess the generality of our findings and determine whether the proposed interventions can be applied beyond the EDM2 framework.

### AI USE STATEMENT

In this work, we used generative AI tools to assist with code development, debugging, and implementation-related tasks, as well as with translation and language polishing of the manuscript. We have not used generative AI tools to conduct experiments, generate experimental results, or make scientific claims, and these uses are not applicable to this work. All AI-assisted code was manually reviewed, tested, and verified for correctness. All AI-assisted translations and language edits were carefully reviewed and revised by the authors to ensure that the intended technical meaning was preserved. We take full responsibility for the final content of this work, including all text, claims, experimental results, and artifacts produced with the aid of generative AI.

### REPRODUCIBILITY STATEMENT

To facilitate reproducibility, the source code used in our experiments will be provided as supplementary material. The complete experimental configurations, including model architectures, training settings, and optimization hyperparameters, are documented in the appendix. Detailed descriptions of the theoretical analysis and robustness experiments are also provided in the appendix. In addition, the datasets used in our experiments and their corresponding preprocessing procedures are described in Appendix A.1. Together, the released source code and the detailed descriptions in the appendix provide the necessary information to reproduce our experimental results and verify our findings.

## REFERENCES

Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. OpenAI Technical Report, 2024. URL https://openai.com/index/video-generation-models-as-world-simulators/.

Shoufa Chen, Chongjian Ge, Yuqi Zhang, Yida Zhang, Fengda Zhu, Hao Yang, Hongxiang Hao, Hui Wu, Zhichao Lai, Yifei Hu, Ting-Che Lin, Shilong Zhang, Fu Li, Chuan Li, Xing Wang, Yanghua Peng, Peize Sun, Ping Luo, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Xiaobing Liu. Goku: Flow based video generative foundation models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pp. 23516–23527. Computer Vision Foundation / IEEE, 2025. doi: 10.1109/CVPR52734.2025.02190. URL https://openaccess.thecvf.com/content/CVPR2025/html/Chen_Goku_Flow_Based_Video_Generative_Foundation_Models_CVPR_2025_paper.html.

Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 8185–8194. Computer Vision Foundation / IEEE, 2020. doi: 10.1109/CVPR42600.2020.00821. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Choi_StarGAN_v2_Diverse_Image_Synthesis_for_Multiple_Domains_CVPR_2020_paper.html.

Jeremy Cohen, Simran Kaur, Yuanzhi Li, J. Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=jh-rTtvkGeM.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pp. 248–255. IEEE Computer Society, 2009. doi: 10.1109/CVPR.2009.5206848. URL https://doi.org/10.1109/CVPR.2009.5206848.

Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat gans on image synthesis. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 8780–8794, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/49ad23d1ec9fa4bd8d77d02681df5cfa-Abstract.html.

Tim Dockhorn, Arash Vahdat, and Karsten Kreis. Score-based generative modeling with critically-damped langevin diffusion. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=CzceR82CYc.

Yonatan Dukler, Quanquan Gu, and Guido Montúfar. Optimization theory for relu neural networks trained with normalization layers. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2751–2760. PMLR, 2020. URL http://proceedings.mlr.press/v119/dukler20a.html.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/6d13e085b79d454da5910e4ca82a3d9d-Abstract-Conference.html.

Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2232–2241. PMLR, 2019. URL http://proceedings.mlr.press/v97/ghorbani19b.html.

Gene H. Golub and Dianne P. O'Leary. Some history of the conjugate gradient and lanczos algorithms: 1948-1976. *SIAM Rev.*, 31(1):50–102, 1989. doi: 10.1137/1031003. URL https://doi.org/10.1137/1031003.

Yuwei Guo, Ceyuan Yang, Anyi Rao, Zhengyang Liang, Yaohui Wang, Yu Qiao, Maneesh Agrawala, Dahua Lin, and Bo Dai. Animatediff: Animate your personalized text-to-image diffusion models without specific tuning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=Fx2SbBgcte.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 6626–6637, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/8a1d694707eb0fefe65871369074926d-Abstract.html.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey A. Gritsenko, Diederik P. Kingma, Ben Poole, Mohammad Norouzi, David J. Fleet, and Tim Salimans. Imagen video: High definition video generation with diffusion models. *CoRR*, abs/2210.02303, 2022. doi: 10.48550/ARXIV.2210.02303. URL https://doi.org/10.48550/arXiv.2210.02303.

Zahra Kadkhodaie, Florentin Guth, Eero P. Simoncelli, and Stéphane Mallat. Generalization in diffusion models arises from geometry-adaptive harmonic representations. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=ANvmVS2Yr0.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=Hk99zCeAb.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 4401–4410. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00453. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Karras_A_Style-Based_Generator_Architecture_for_Generative_Adversarial_Networks_CVPR_2019_paper.html.

Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/8d30aa96e72440759f74bd2306c1fa3d-Abstract.html.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/a98846e9d9cc01cfb87eb694d946ce6b-Abstract-Conference.html.

Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and improving the training dynamics of diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pp. 24174–24184. IEEE, 2024. doi: 10.1109/CVPR52733.2024.02282. URL https://doi.org/10.1109/CVPR52733.2024.02282.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 7482–7491. Computer Vision Foundation / IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00781. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Kendall_Multi-Task_Learning_Using_CVPR_2018_paper.html.

Beomsu Kim, Yu-Guan Hsieh, Michal Klein, Marco Cuturi, Jong Chul Ye, Bahjat Kawar, and James Thornton. Simple reflow: Improved techniques for fast flow models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=fpvgSDKXGY.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Weijie Kong, Qi Tian, and Zijian Zhang. Hunyuanvideo: A systematic framework for large video generation models. 2025. URL https://arxiv.org/abs/2412.03603.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=a-xFK8Ymz5J.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. *CoRR*, abs/2511.13720, 2025. doi: 10.48550/ARXIV.2511.13720. URL https://doi.org/10.48550/arXiv.2511.13720.

Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pp. 300–309. IEEE, 2023. doi: 10.1109/CVPR52729.2023.00037. URL https://doi.org/10.1109/CVPR52729.2023.00037.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=XVjTT1nw5z.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8162–8171. PMLR, 2021. URL http://proceedings.mlr.press/v139/nichol21a.html.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/48b383b24230e0e6e649d9c98dae4d8c-Abstract-Conference.html.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pp. 4172–4182. IEEE, 2023. doi: 10.1109/ICCV51070.2023.00387. URL https://doi.org/10.1109/ICCV51070.2023.00387.

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=FjNys5c7VyY.

Vadim Popov, Ivan Vovk, Vladimir Gogoryan, Tasnima Sadekova, and Mikhail A. Kudinov. Grad-tts: A diffusion probabilistic model for text-to-speech. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8599–8608. PMLR, 2021. URL http://proceedings.mlr.press/v139/popov21a.html.

Amit Raj, Srinivas Kaza, Ben Poole, Michael Niemeyer, Nataniel Ruiz, Ben Mildenhall, Shiran Zada, Kfir Aberman, Michael Rubinstein, Jonathan T. Barron, Yuanzhen Li, and Varun Jampani.

Dreambooth3d: Subject-driven text-to-3d generation. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pp. 2349–2359. IEEE, 2023. doi: 10.1109/ICCV51070.2023.00223. URL https://doi.org/10.1109/ICCV51070.2023.00223.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells III, and Alejandro F. Frangi (eds.), *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*, volume 9351 of *Lecture Notes in Computer Science*, pp. 234–241. Springer, 2015. doi: 10.1007/978-3-319-24574-4_28. URL https://doi.org/10.1007/978-3-319-24574-4_28.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=TIdIXIpzhoI.

Tim Salimans and Diederik P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 901, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/ed265bc903a5a097f61d3ec064d96d2e-Abstract.html.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 2488–2498, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/905056c1ac1dad141560467e0a99e1cf-Abstract.html.

J. Ryan Shue, Eric Ryan Chan, Ryan Po, Zachary Ankner, Jiajun Wu, and Gordon Wetzstein. 3d neural field generation using triplane diffusion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pp. 20875–20886. IEEE, 2023. doi: 10.1109/CVPR52729.2023.02000. URL https://doi.org/10.1109/CVPR52729.2023.02000.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis R. Bach and David M. Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 2256–2265. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/sohl-dickstein15.html.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021a. URL https://openreview.net/forum?id=St1giarCHLP.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 11895–11907, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/3001ef257407d5a371a96dcd947c7d93-Abstract.html.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria,*

*May 3-7, 2021*. OpenReview.net, 2021b. URL https://openreview.net/forum?id=PxTIG12RRHS.

Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=UyNXMqnN3c.

Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 11287–11302, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/5dca4c6b9e244d24a30b4c45601d9720-Abstract.html.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Comput.*, 23(7):1661–1674, 2011. doi: 10.1162/NECO_A_00142. URL https://doi.org/10.1162/NECO_a_00142.

Xingyi Yang, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Diffusion probabilistic model made slim. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pp. 22552–22562. IEEE, 2023. doi: 10.1109/CVPR52729.2023.02160. URL https://doi.org/10.1109/CVPR52729.2023.02160.

Aoxiong Yin, Xu Tan, Kai Shen, Yichong Leng, Xinyu Zhou, Juncheng Li, and Siliang Tang. The best of both worlds: Integrating language models and diffusion models for video generation. In *IEEE/CVF International Conference on Computer Vision, ICCV 2025, Honolulu, HI, USA, October 19-25, 2025*, pp. 1–12. IEEE, 2025. doi: 10.1109/ICCV51701.2025.01448. URL https://doi.org/10.1109/ICCV51701.2025.01448.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 12360–12371, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/1e8a19426224ca89e83cef47f1e7f53b-Abstract.html.

Yifan Zhang and Bryan Hooi. Hipa: Enabling one-step text-to-image diffusion models via high-frequency-promoting adaptation. *CoRR*, abs/2311.18158, 2023. doi: 10.48550/ARXIV.2311.18158. URL https://doi.org/10.48550/arXiv.2311.18158.

## A Implementation Details

All methods and ablation studies in this work are implemented based on the official open-source codebases of EDM and EDM2. Unless otherwise specified, all experiments are conducted on NVIDIA A30 GPUs with 24 GB of memory. The software environment consists of Python 3.10.19, PyTorch 2.3.0, CUDA 12.1, and cuDNN 8.9.2. All training runs are performed on a single compute node equipped with four GPUs, and FID evaluation is likewise performed using four GPUs.

### A.1 DATASETS

To ensure fair comparisons with existing methods, all datasets are processed using the official preprocessing pipeline provided by the EDM codebase. We use CIFAR-10, FFHQ-64, and AFHQv2-64 in our experiments. For each dataset, we directly use the corresponding original dataset and apply the standard EDM preprocessing procedure without any additional modification.

### A.2 FID EVALUATION

We follow the evaluation protocol of EDM and compute FID using 50,000 generated samples and the complete training set as the reference set. No data augmentation is applied during evaluation.

For feature extraction, we use the 'InceptionV3Detector' provided by the official EDM2 implementation. This detector loads the 'inception-2015-12-05.pkl' Inception-v3 model distributed with the StyleGAN3 metrics package and extracts its 2,048-dimensional features for FID computation. All sampling and evaluation procedures use a fixed global random seed to ensure consistent evaluation across experiments.

For each evaluated checkpoint, we additionally perform a post-hoc sweep over the EMA length used in the EDM2 implementation, ranging from $0.010$ to $0.200$ with a step size of $0.005$. The reported FID corresponds to the best value obtained among these post-hoc EMA settings.

### A.3 SAMPLING PROCEDURE

All samples are generated using the second-order deterministic sampler proposed by Karras et al. (2022), and corresponding to Algorithm 1 in EDM. Unless otherwise specified, we use

$$\sigma_{\min} = 0.002, \qquad \sigma_{\max} = 80, \qquad \rho = 7. \tag{8}$$

The noise schedule and scaling function are set to

$$\sigma(t) = t, \qquad s(t) = 1. \tag{9}$$

We use 18 sampling steps for CIFAR-10 and 40 sampling steps for FFHQ-64 and AFHQv2-64, following the corresponding EDM configurations. Classifier-free guidance (Ho & Salimans, 2021) is disabled in all experiments.

### A.4 MIXED-PRECISION COMPUTATION

All training procedures follow the mixed-precision computation scheme of EDM2. Trainable parameters are maintained in FP32 throughout training, while computationally intensive operations in the forward pass are performed in FP16 whenever numerically stable. The embedding network and its associated linear layers are kept in FP32, whereas other activation tensors are stored and processed in FP16. With appropriate loss scaling, we observe no noticeable difference in convergence behavior between mixed-precision and full-FP32 training.

### A.5 ACTIVATION STATISTICS

To clarify the activation-related terminology used throughout the paper, we define the activation mean and root mean square (RMS) statistics used in our analysis. Given a set of 4,096 training samples, we compute the statistics over all activation elements produced by the corresponding network layers. For clarity, we omit the mini-batch dimension and denote an activation tensor for a single sample by $\mathbf{h} \in \mathbb{R}^{N \times H \times W}$, where $N$ is the number of feature channels and $H$ and $W$ denote the spatial dimensions. We reshape the activation tensor as $\mathbf{h} \in \mathbb{R}^{N \times M}$, where $M = HW$.

**Activation Mean.** The global mean of an activation tensor is defined as

$$\mu(\mathbf{h}) = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} h_{ij}. \tag{10}$$

When computed over the 4,096 training samples, the reported statistic corresponds to the empirical expectation of the activation values over all samples, channels, and spatial locations.

**Activation RMS.** Throughout the paper, activation magnitude and signal RMS refer to the same quantity. For an activation tensor $\mathbf{h}$, we define its magnitude as the root mean square of all activation values:

$$\mathrm{RMS}(\mathbf{h}) = \sqrt{\frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M} h_{ij}^2}. \tag{11}$$

# B Technical Details

This section provides implementation details of the methods introduced in Section 2. Unless otherwise specified, all descriptions and architectural illustrations in this appendix refer to the CIFAR-10 configuration. Figure 5, Figure 6, and Figure 7 illustrate the network architectures corresponding to Configurations A, B, and D, respectively, together with their associated hyperparameter settings. For FFHQ-64 and AFHQv2-64, we use the EDM2 architecture with the network configuration and training hyperparameters adopted from the original EDM. Any modifications specific to our method or these datasets are described separately where applicable.

We recommend referring to the electronic version of the paper for high-resolution architecture diagrams.

## B.1 Baseline Architecture (Configuration A)

Our baseline model is built upon the EDM2 architecture proposed by Karras et al. (2024) and implemented within the EDM training framework. EDM2 is developed from the ADM U-Net architecture (Dhariwal & Nichol, 2021) through a series of architectural and parameterization modifications. Since EDM2 only reports experiments on ImageNet (Deng et al., 2009)-64 and ImageNet-512, we make several dataset-specific configuration adjustments when transferring the architecture to smaller-scale image generation benchmarks, including CIFAR-10, FFHQ-64, and AFHQv2-64.

This section focuses on the architectural and implementation details that are directly relevant to our experiments. For a complete description of the EDM2 architecture and its design motivations, we refer readers to Karras et al. (2024).

Unless otherwise specified, all training and sampling hyperparameters follow the original EDM and EDM2 configurations, except for the LR ramp-up duration. The configuration differences among CIFAR-10, FFHQ-64, and AFHQv2-64 are summarized in Table 3. The complete network architecture and configuration of the CIFAR-10 baseline are illustrated in Figure 5.

Table 3: Training configuration of Configuration A across CIFAR-10, FFHQ-64, and AFHQv2-64.

| **Configuration** | **CIFAR-10** | **AFHQv2-64** | **FFHQ-64** |
|---|---|---|---|
| Learning rate max | 0.017 | 0.017 | 0.017 |
| Learning rate decay | 20000 | 20000 | 20000 |
| Minibatch size | 512 | 256 | 256 |
| LR ramp-up (Mimg) | 1 | 1.5 | 1.5 |
| Dropout probability | 13% | 25% | 5% |
| Channel multiplier | 128 | 128 | 128 |
| Channels per resolution | 2-2-2 | 1-2-2-2 | 1-2-2-2 |
| Duration(Mimg) | 50 | 60 | 65 |
| EMA length | 0.010 | 0.010 | 0.010 |

### B.1.1 EDM Preconditioning

Following the EDM formulation, the network is trained to predict the denoised estimate:

$$\tilde{y} = D_\theta(x; \sigma, c), \tag{12}$$

where

$$x = y + n \tag{13}$$

denotes the noisy input, $y$ is the clean image, $n \sim \mathcal{N}(0, \sigma^2 I)$ is Gaussian noise, $\sigma$ represents the noise level, and $c$ denotes the class label.

All CIFAR-10 images are normalized into the range $[-1, 1]$. The processed dataset has approximately zero mean with

$$\sigma_{\text{data}} = 0.5. \tag{14}$$

Instead of directly learning $D_\theta$, EDM decomposes the denoiser into fixed preconditioning functions and a trainable neural network $F_\theta$:

$$D_\theta(x; \sigma, c) = C_{\text{skip}}(\sigma)x + C_{\text{out}}(\sigma)F_\theta\left(C_{\text{in}}(\sigma)x, C_{\text{noise}}(\sigma), c\right). \tag{15}$$

The coefficients $C_{\text{in}}$, $C_{\text{out}}$, $C_{\text{skip}}$, and $C_{\text{noise}}$ are designed such that the input and output signal magnitudes of $F_\theta$ remain approximately normalized across different noise levels. The complete derivation follows EDM and is omitted here.

EDM constructs $D_\theta$ by drawing on earlier parameterization approaches (Dockhorn et al., 2022; Salimans & Ho, 2022; Vahdat et al., 2021) that adaptively combine signal and noise. With the explicit skip connection from the input to the output, $D_\theta$ can be viewed as a residual formulation in which the trainable network $F_\theta$ learns only a correction to the input signal. This design is conceptually consistent with the direct denoising perspective of Li & He (2025), which shows that directly predicting clean data can achieve high-quality generation.

### B.1.2 Network Architecture

The overall backbone follows the ADM U-Net architecture adopted by EDM2, with self-attention (Vaswani et al., 2017) blocks.

The noisy input is first multiplied by $C_{\text{in}}$ and projected into the feature space through the input layer ("In" module in Figure 5). The resulting activations are processed by hierarchical encoder and decoder blocks connected through standard U-Net skip connections.

For CIFAR-10, the encoder progressively reduces the spatial resolution from $32 \times 32$ to $16 \times 16$ and $8 \times 8$, while the decoder restores the resolution in the reverse order. Unlike conventional U-Net implementations, we follow EDM and maintain a constant channel width throughout the network.

The final output layer ("Out") projects the feature representation back to RGB space, after which EDM preconditioning is applied to obtain the final denoised prediction.

To ensure a fair comparison with the SongUNet architecture used by EDM on CIFAR-10, we retain attention at the $16 \times 16$ decoder resolution and in the final decoder block, while removing the other decoder attention blocks. All other architectural settings follow EDM2.

### B.1.3 Conditional Modulation

Since EDM2 removes all GroupNorm layers, it no longer uses normalization-based conditioning mechanisms. Instead, noise-level and class conditions are embedded into a shared conditioning vector, which modulates residual branch activations through RMS-preserving scaling.

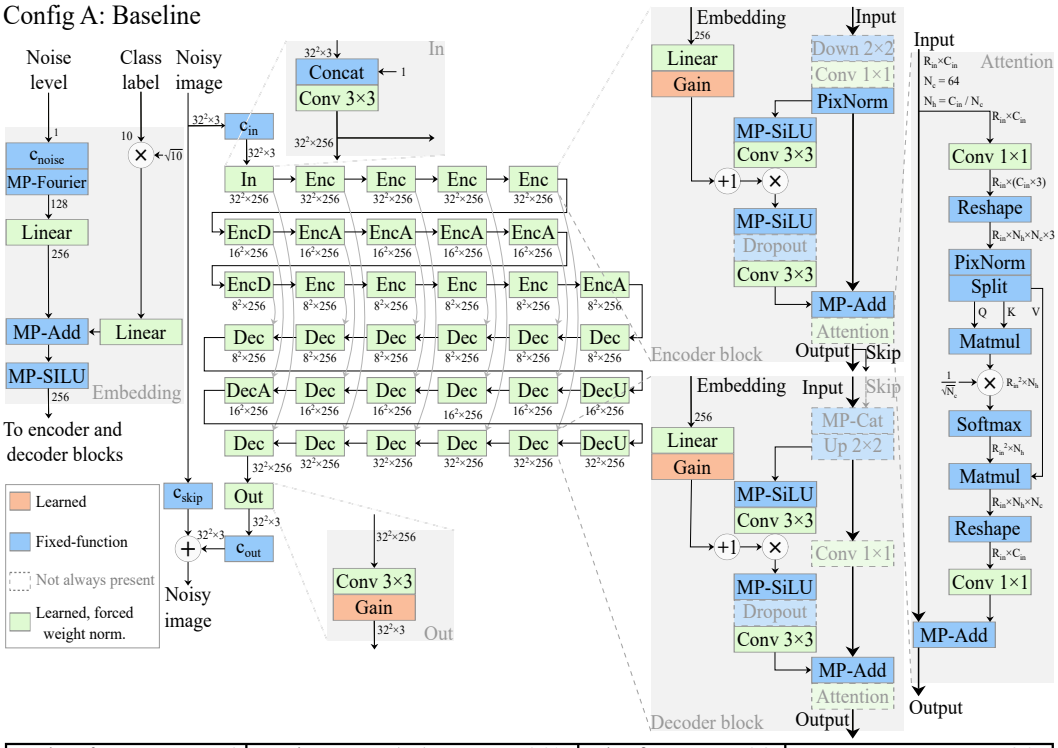
Specifically, the noise level $\sigma$ is first transformed using magnitude-preserving Fourier embedding (MP-Fourier) and then passed through a linear layer. The class label is represented using one-hot encoding, multiplied by a scaling factor to achieve unit RMS in expectation, and then projected through a linear layer. The two embeddings are combined using magnitude-preserving summation (MP-Sum) and processed by a multi-layer embedding network consisting of linear layers and magnitude-preserving SiLU activations (MP-SiLU). The resulting condition embedding is transformed into a channel-wise modulation vector:

$$s = \alpha_e W_e e + 1, \tag{16}$$

where $e$ denotes the condition embedding, $W_e$ is a learnable linear projection, and $\alpha_e$ is a learnable scalar initialized to zero. The residual activation tensor $h$ is then modulated as:

$$h \leftarrow h \odot s. \tag{17}$$

Following EDM2, the modulation layer is initialized with zero gain, ensuring that conditioning is initially inactive at the beginning of training.




| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Number of GPUs | 4 | Learning rate max($\alpha_{ref}$) | 0.017 | Adam $\beta_1$ | 0.9 | FID | 2.05 |
| Minibatch size | 512 | Learning rate decay($t_{ref}$) | 20000 | Adam $\beta_2$ | 0.99 | EMA length($\sigma_{rel}$) | 0.010 |
| Duration(Mimg) | 50 | Learning rate rampup(Mimg) | 1.0 | Loss scaling | 1 | Model capacity(Mparams) | 5.4 |
| Mixed-precision(FP16) | full | Noise distribution mean($P_{mean}$) | -1.2 | Attention res | 16 | Channel multiplier | 128 |
| Dropoutprobability | 0.13 | Noise distribution std. ($P_{std}$) | 1.2 | Attention blocks | 6 | Channels per resolution | 2-2-2 |

Figure 5: Network architecture and configuration of Configuration A on CIFAR-10.

### B.1.4 Residual Blocks and Attention Layers

The encoder and decoder consist of magnitude-preserving residual blocks. Each residual branch contains two convolutional layers, with MP-SiLU activations applied before each convolution. Residual merging is implemented using MP-Sum to maintain stable activation magnitudes after residual addition.

During training, dropout is applied after the second convolution layer. Unless otherwise specified, all convolutional and linear layers are initialized following the EDM2 initialization scheme, where weights are sampled from a standard normal distribution.

Self-attention is enabled only at the $16 \times 16$ spatial resolution, following the original EDM configuration. The attention implementation follows the standard multi-head attention formulation. Queries, keys, and values are generated using $1 \times 1$ convolutions, and each attention head has a dimension of 64.

Unlike conventional attention mechanisms based on raw dot-product similarity, EDM2 applies pixel normalization to query, key, and value vectors before computing attention logits. Therefore, the attention scores depend only on vector directions rather than magnitudes, resulting in cosine-similarity attention.

### B.1.5 Magnitude-Preserving Components

One of the major architectural differences introduced by EDM2 is the replacement of Group Normalization with magnitude-preserving components. The objective of these designs is to maintain stable signal magnitudes throughout the network during training.

**Weight Normalization.** EDM2 replaces activation normalization with weight normalization. For a linear layer

$$y = Wx, \tag{18}$$

each output channel corresponds to a weight vector $w_i$. EDM2 normalizes each weight vector as:

$$\hat{w}_i = \frac{w_i}{\|w_i\|_2}. \tag{19}$$

Under the standard independence assumptions used in signal propagation analysis, this operation approximately preserves the variance of the output signal with respect to the input signal. Therefore, the signal magnitude remains stable when passing through learnable layers.

However, conventional weight normalization introduces an optimization issue. Since gradients are projected onto the tangent space of the weight sphere, the magnitude of the unconstrained parameter vector may gradually increase during training. This implicit growth effectively reduces the learning rate of the normalized weight direction, making the optimization dynamics depend on parameter norms rather than solely on the explicit learning-rate schedule.

To eliminate this effect, EDM2 introduces forced weight normalization. Before each forward pass, every weight vector is explicitly projected onto a sphere with a fixed radius:

$$\hat{w}_i \leftarrow \sqrt{d} \cdot \frac{w_i}{\|w_i\|_2} \tag{20}$$

where $d$ denotes the dimensionality of the weight vector. As a result, the weight RMS remains constant throughout training, and the effective learning rate is controlled only by the explicit learning-rate schedule.

**Magnitude-Preserving Operations.** A central design principle of EDM2 is maintaining approximately constant signal magnitudes throughout the network. To achieve this goal, several commonly used operations are replaced with magnitude-preserving (MP) variants, including MP-Sum, MP-Cat, MP-SiLU, and MP-Fourier.

These operations are designed based on signal propagation analysis such that the RMS magnitude of activations remains approximately unchanged before and after the corresponding transformation. Detailed derivations are provided in the EDM2 paper. For completeness, we describe MP-Sum and MP-Cat, which are particularly relevant to our later analysis.

For residual addition, EDM2 replaces standard summation with MP-Sum:

$$\text{MP-Sum}(a, b) = \frac{(1-t)a + tb}{\sqrt{(1-t)^2 + t^2}}, \tag{21}$$

where $t$ controls the relative contribution of the two input branches. Under the assumption that the two inputs have similar RMS magnitudes, the denominator is designed to approximately preserve the RMS magnitude before and after the transformation.

For feature concatenation, EDM2 introduces MP-Cat:

$$\text{MP-Cat}(a, b) = \frac{\sqrt{N_a + N_b}}{\sqrt{(1-t)^2 + t^2}} \left( \frac{1-t}{\sqrt{N_a}} a \oplus \frac{t}{\sqrt{N_b}} b \right), \tag{22}$$

where $N_a$ and $N_b$ denote the channel dimensions of the two input feature maps, and $\oplus$ represents channel-wise concatenation.

Unlike MP-Sum, concatenation itself does not directly alter the RMS magnitude of individual feature values. Instead, MP-Cat adjusts the relative scaling of the two feature streams according to their channel dimensions, thereby controlling their relative signal energy after concatenation. EDM2 fixes $t = 0.5$, under which the two branches contribute equally to the total signal energy. In the special case where $t = 0.5$ and $N_a = N_b$, MP-Cat reduces to standard channel-wise concatenation and introduces no additional scaling.

The remaining MP-SiLU and MP-Fourier operations follow the same principle of magnitude preservation. Instead of introducing additional normalization layers, they compensate for the magnitude change caused by nonlinear transformations through fixed scaling factors.

For MP-SiLU, the SiLU activation changes the expected signal magnitude because its output distribution no longer follows the same RMS statistics as its input. Therefore, EDM2 rescales the activation output by a fixed factor:

$$\text{MP-SiLU}(x) = \frac{\text{SiLU}(x)}{\alpha_{\text{SiLU}}}, \tag{23}$$

where $\alpha_{\text{SiLU}} = 0.596$ is selected according to the expected RMS of the SiLU output.

Similarly, MP-Fourier modifies the standard Fourier embedding by applying a magnitude correction factor:

$$\text{MP-Fourier}(x) = \sqrt{2} \cdot \text{Fourier}(x), \tag{24}$$

where the scaling factor $\sqrt{2}$ compensates for the RMS reduction introduced by the cosine projection.

Both operations preserve the signal magnitude without introducing learnable normalization parameters. The detailed derivations of these scaling factors follow EDM2 and are omitted for brevity.

#### B.1.6 Uncertainty-based Multi-task Training

EDM2 additionally employs an uncertainty-based multi-task training strategy. Instead of manually assigning fixed weights to different loss terms, it introduces learnable uncertainty parameters to dynamically balance multiple objectives.

Following Kendall et al. (2018), the combined objective for multiple tasks can be formulated as:

$$\mathcal{L} = \sum_i \left( \frac{1}{2\sigma_i^2} \mathcal{L}_i + \log \sigma_i \right), \tag{25}$$

where $\sigma_i$ represents the learned homoscedastic uncertainty associated with the $i$-th task.

EDM2 applies this formulation to diffusion training by parameterizing the uncertainty as a function of noise level. Specifically, the objective is rewritten as:

$$\mathcal{L}(D_\theta, u) = \mathbb{E}_\sigma \left[ \frac{\lambda(\sigma)}{e^{u(\sigma)}} \mathcal{L}(D_\theta; \sigma) + u(\sigma) \right], \tag{26}$$

where $\lambda(\sigma)$ denotes the EDM weighting function and $e^{-u(\sigma)}$ serves as the learned uncertainty-based task weighting. Importantly, $u(\sigma)$ is a network-predicted log-variance conditioned on the diffusion noise level $\sigma$ and should not be confused with $\ln \sigma^2$ or with the diffusion noise level itself.

This formulation allows the relative contribution of the reconstruction objective to adapt automatically according to the diffusion noise level, without requiring a manually specified uncertainty weight. Unless otherwise specified, our experiments retain this uncertainty-based multitask objective following EDM2.

#### B.1.7 Other Architectural Modifications in EDM2

Besides the magnitude-preserving components, cosine attention mechanism, and uncertainty-based multi-task loss introduced above, EDM2 incorporates several additional architectural and optimization refinements compared with the original EDM framework. Although these modifications operate independently of the magnitude-preservation scheme, they play a crucial role in enhancing training stability. We detail these differences below to clarify the precise components isolated in our progressive ablation study (Section 2.5).

**Resolution-changing operation ordering.** EDM2 modifies the ordering of resolution-changing operations inside residual blocks. In the original EDM implementation, the feature stream is first split into the main branch and residual branch, after which the two branches perform resolution changes independently. EDM2 instead applies the resolution-changing operation before splitting the feature stream. This design ensures that both branches operate on feature representations with consistent spatial resolutions and avoids potential inconsistencies introduced by performing separate resampling operations.

**Fourier noise embedding and multiplicative conditioning.** EDM2 replaces the sinusoidal noise-level embedding used in EDM with Fourier feature embedding. In addition, the embedding network

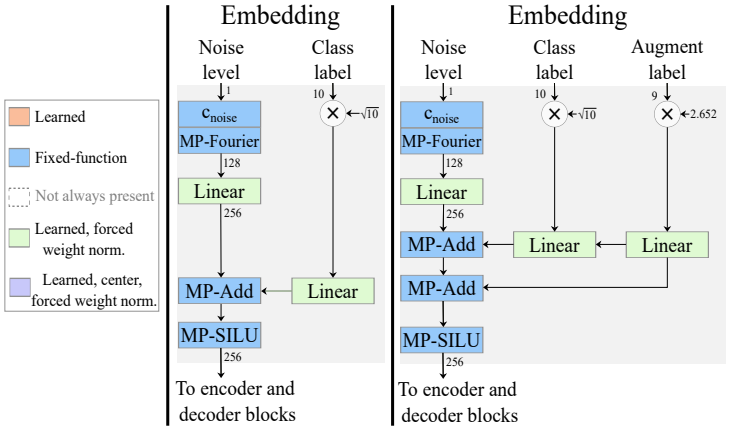



Figure 6: Comparison of the conditioning pathways between Configuration A and Configuration B. The two configurations share the same EDM2 backbone and training components. Only the embedding pathway is modified: Configuration A directly combines the noise-level and class embeddings, while Configuration B additionally introduces the normalized augmentation embedding through MP-Sum.

is simplified by removing one SiLU-linear transformation block after the initial embedding projection. The conditioning mechanism is also modified. Instead of additive bias injection, EDM2 applies multiplicative modulation to residual activations. Specifically, while EDM uses an additive transformation:

$$h \leftarrow h + b(e), \tag{27}$$

where $e$ denotes the condition embedding, EDM2 adopts:

$$h \leftarrow h \odot (1 + g(e)), \tag{28}$$

where $g(e)$ is a learned channel-wise modulation vector. This formulation is consistent with the magnitude-preserving design philosophy of EDM2 and allows conditioning information to directly modulate feature magnitudes.

**Zero-initialized output and embedding gains.** EDM2 introduces two additional learnable scalar parameters, out_gain and emb_gain, which are initialized to zero. The output gain controls the final residual network output:

$$F_\theta(x) \leftarrow \text{out_gain} \cdot F_\theta(x), \tag{29}$$

so that the residual prediction branch is initially inactive. Consequently, the complete denoiser starts close to the identity mapping, which improves optimization stability at the beginning of training. Similarly, emb_gain controls the strength of conditional modulation:

$$g(e) \leftarrow \text{emb_gain} \cdot g(e), \tag{30}$$

ensuring that conditioning is gradually introduced during training rather than being active from initialization.

**Post-hoc EMA.** EDM2 introduces a post-hoc EMA strategy to provide flexible control over the effective EMA length. During training, EDM2 maintains two EMA trajectories with fixed decay coefficients. Periodically, snapshots of these EMA parameters are stored throughout training. After training completes, these snapshots are combined to reconstruct models corresponding to different effective EMA lengths. Compared with conventional EMA, where the optimal decay coefficient must be selected before or during training, post-hoc EMA enables the EMA smoothing strength to be adjusted after training. This provides a more flexible way to balance optimization responsiveness and parameter smoothing, since different EMA lengths can be evaluated without retraining the model. The reconstruction procedure follows the formulation introduced in EDM2 and is omitted here for brevity.

### B.2 NON-LEAKING DATA AUGMENTATION (CONFIGURATION B)

The non-leaking augmentation strategy was originally introduced by Karras et al. (2020), for generative adversarial networks and later adopted in the EDM framework. The augmentation pipeline applies a set of stochastic geometric transformations, including flipping, rotation, translation, and scaling. The complete augmentation configuration is summarized in Table 4.

Directly applying augmentation without providing the transformation information to the network may introduce a mismatch between the training and inference distributions. To prevent the model from interpreting augmentation artifacts as intrinsic image structures, EDM encodes the augmentation parameters and provides them as additional conditioning information.

Let $\mathbf{a} \in \mathbb{R}^9$ denote the augmentation parameter vector. Unlike the original EDM implementation, we modify the augmentation conditioning pathway to follow the magnitude-preserving design of EDM2. Specifically, the augmentation parameters are first scaled by a fixed normalization factor derived from their expected RMS magnitude:

$$\hat{\mathbf{a}} = \frac{\mathbf{a}}{\mathrm{RMS}(\mathbf{a})}. \tag{31}$$

The normalized augmentation embedding is then projected through a linear layer:

$$\mathbf{e}_{\mathrm{aug}} = W_{\mathrm{aug}}\hat{\mathbf{a}}. \tag{32}$$

The resulting augmentation embedding is incorporated into the shared conditioning pathway through MP-Sum:

$$\mathbf{e} = \text{MP-Sum}\left(\text{MP-Sum}\left(\mathbf{e}_\sigma, \mathbf{e}_c, 0.5\right), \mathbf{e}_{\mathrm{aug}}, 0.3\right), \tag{33}$$

where $\alpha = 0.3$ controls the relative contribution of augmentation conditioning.

A large value of $\alpha$ may cause augmentation information to dominate the shared condition vector, reducing the influence of diffusion-time and class information. Conversely, an excessively small value weakens the effectiveness of augmentation conditioning.

Following the original augmentation parameterization, the RMS magnitude of the raw augmentation vector can be analytically estimated. Let $p = A_{\mathrm{prob}}$ denote the probability of applying stochastic transformations. Because each augmentation component is independently sampled and becomes zero when the corresponding transformation is disabled, the element-wise RMS is given by:

$$\mathrm{RMS}(\mathbf{a}) = \sqrt{\frac{1}{9}\sum_{i=1}^{9}\mathbb{E}[a_i^2]}. \tag{34}$$

Using the augmentation distributions listed in Table 4, the second moments of the nine components are:

$$\{0.5,\ 0.5p,\ p,\ 1.5p,\ 0.5p,\ 0.5p,\ 0.5p,\ p,\ p\}\,. \tag{35}$$

Therefore,

$$\sum_{i=1}^{9}\mathbb{E}[a_i^2] = 0.5 + 6.5p, \tag{36}$$

and the corresponding RMS magnitude is:

$$\mathrm{RMS}(\mathbf{a}) = \sqrt{\frac{0.5 + 6.5A_{\mathrm{prob}}}{9}}. \tag{37}$$

For CIFAR-10, where $A_{\mathrm{prob}} = 0.12$, the resulting RMS magnitude is approximately $0.377$. For FFHQ-64 and AFHQv2-64, where $A_{\mathrm{prob}} = 0.15$, the RMS magnitude is approximately $0.405$. These values are used to normalize the augmentation condition before magnitude-preserving fusion. This normalization ensures that the augmentation condition has a comparable magnitude to the existing EDM2 conditioning signals before magnitude-preserving fusion. The difference between the conditioning pathways in Configuration A and Configuration B is illustrated in Figure 6.The detailed configurations of Configuration B for CIFAR-10, FFHQ-64, and AFHQv2-64 are provided in Table 5; all unspecified parameters remain the same as those in Configuration A.

Table 4: The augmentation pipeline adopted from EDM (Karras et al., 2022) and applied to EDM2 in our experiments. Each training image undergoes a combined geometric transformation based on 8 random parameters, which receive non-zero values with a certain probability. The model is conditioned on an additional 9-dimensional input vector derived from these augmentation parameters.

| Augmentation | Transformation | Parameters | Prob. | Conditioning | Constants |
|---|---|---|---|---|---|
| $x$-flip<br>$y$-flip | SCALE2D$(1-2a_0,\ 1)$<br>SCALE2D$(1,\ 1-2a_1)$ | $a_0 \sim \mathcal{U}\{0,1\}$<br>$a_1 \sim \mathcal{U}\{0,1\}$ | 100%<br>$A_{\text{prob}}$ | $a_0$<br>$a_1$ | $A_{\text{prob}} = 12\%$<br>or 15% |
| Scaling | SCALE2D $((A_{\text{scale}})^{a_2},$<br>$(A_{\text{scale}})^{a_2})$ | $a_2 \sim \mathcal{N}(0,1)$ | $A_{\text{prob}}$ | $a_2$ | $A_{\text{scale}} = 2^{0.2}$ |
| Rotation | ROTATE2D$(-a_3)$ | $a_3 \sim \mathcal{U}(-\pi,\pi)$ | $A_{\text{prob}}$ | $\cos a_3 - 1$<br>$\sin a_3$ | |
| Anisotropy | ROTATE2D$(a_4)$<br>SCALE2D $((A_{\text{aniso}})^{a_5},$<br>$1/(A_{\text{aniso}})^{a_5})$<br>ROTATE2D$(-a_4)$ | $a_4 \sim \mathcal{U}(-\pi,\pi)$<br>$a_5 \sim \mathcal{N}(0,1)$ | $A_{\text{prob}}$ | $a_5 \cos a_4$<br>$a_5 \sin a_4$ | $A_{\text{aniso}} = 2^{0.2}$ |
| Translation | TRANSLATE2D $((A_{\text{trans}})a_6,$<br>$(A_{\text{trans}})a_7)$ | $a_6 \sim \mathcal{N}(0,1)$<br>$a_7 \sim \mathcal{N}(0,1)$ | $A_{\text{prob}}$ | $a_6$<br>$a_7$ | $A_{\text{trans}} = 1/8$ |

Table 5: Training configurations used for the Configuration B on different datasets.

| **Configuration** | **CIFAR-10** | **AFHQv2-64** | **FFHQ-64** |
|---|---|---|---|
| Learning rate max | 0.017 | 0.017 | 0.017 |
| Learning rate decay | 20000 | 20000 | 20000 |
| Minibatch size | 512 | 256 | 256 |
| LR ramp-up (Mimg) | 1 | 1.5 | 1.5 |
| Dropout probability | 13% | 25% | 5% |
| Channel multiplier | 128 | 128 | 128 |
| Channels per resolution | 2-2-2 | 1-2-2-2 | 1-2-2-2 |
| Duration(Mimg) | 80 | 65 | 75 |
| EMA length | 0.015 | 0.010 | 0.010 |
| Augment probability | 12% | 15% | 15% |

### B.3 Piecewise Learning-Rate Annealing (Configuration C)

We found that the inverse-square-root learning-rate schedule adopted by EDM2 is not fully suitable for the small-scale image generation benchmarks considered in this work. In particular, a relatively large initial learning rate enables the model to rapidly enter a low-FID regime, but the optimal EMA decay remains relatively small during this stage. This suggests that the parameter trajectory still changes noticeably during optimization, motivating us to investigate whether stronger learning-rate decay can improve late-stage optimization stability.

Additional experiments with different learning-rate settings reveal that reducing the initial learning rate improves the stability of optimization but prevents the model from reaching the same FID within the same training budget. Therefore, maintaining a sufficiently large learning rate during the early training stage is important for achieving rapid convergence.

These observations suggest that the optimization process benefits from two different stages: an early exploration stage, where relatively large updates enable rapid progress toward high-quality solutions, followed by a refinement stage, where smaller updates help stabilize the final solution.

The inverse-square-root schedule used by EDM2 continuously decreases the learning rate throughout training, but on CIFAR-10 its transition from the exploration stage to the refinement stage is too gradual. We therefore introduce an explicit annealing stage before the onset of the late-training FID degradation observed under the original schedule.

The resulting learning-rate schedule is defined as:

$$\eta(t) = \begin{cases} \eta_{\mathrm{EDM2}}(t), & t < t_{\mathrm{anneal}}, \\ \eta_1, & t_1 \le t < t_2, \\ \eta_2, & t_2 \le t < t_3, \\ \dots & \\ \eta_K, & t \ge t_K, \end{cases} \tag{38}$$

where $t$ denotes the number of training images. Before the annealing stage, we retain the original EDM2 warm-up and inverse-square-root learning-rate schedule. Specifically, we use a reference learning rate of $0.017$, a reference batch count of $20{,}000$, and a warm-up duration of $1$M images. After $35.7$M training images, the learning rate is explicitly reduced according to the following piecewise schedules.

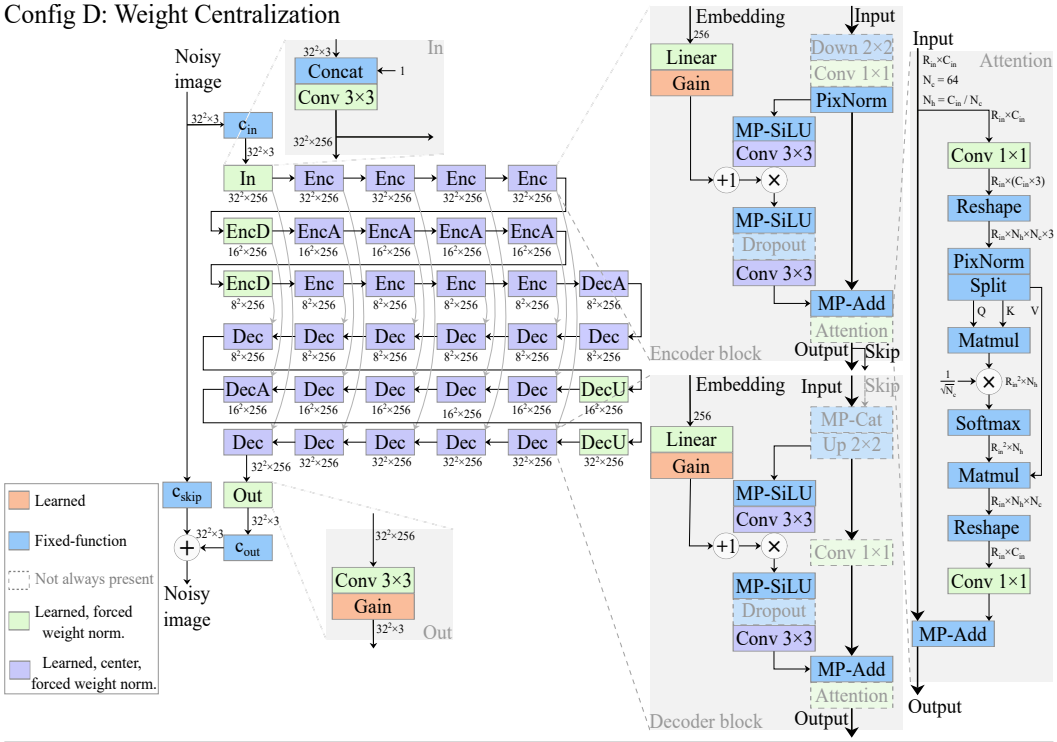
For CIFAR-10, the learning-rate schedule is

$$\eta_{\mathrm{CIFAR}}(t) = \begin{cases} 0.0075, & 35.7\mathrm{M} \le t < 37.7\mathrm{M}, \\ 0.0050, & 37.7\mathrm{M} \le t < 42.9\mathrm{M}, \\ 0.0030, & 42.9\mathrm{M} \le t < 50.0\mathrm{M}, \\ 0.0021, & 50.0\mathrm{M} \le t < 53.9\mathrm{M}, \\ 0.0018, & 53.9\mathrm{M} \le t < 63.9\mathrm{M}, \\ 0.0016, & 63.9\mathrm{M} \le t < 78.9\mathrm{M}, \\ 0.00155, & 78.9\mathrm{M} \le t < 88.9\mathrm{M}, \\ 0.0014, & 88.9\mathrm{M} \le t < 108.9\mathrm{M}, \\ 0.00135, & 108.9\mathrm{M} \le t < 118.9\mathrm{M}, \\ 0.00130, & 118.9\mathrm{M} \le t < 128.9\mathrm{M}, \\ 0.00125, & 128.9\mathrm{M} \le t < 138.9\mathrm{M}, \\ 0.00120, & 138.9\mathrm{M} \le t < 148.9\mathrm{M}, \\ 0.00115, & 148.9\mathrm{M} \le t < 158.9\mathrm{M}, \\ 0.00110, & 158.9\mathrm{M} \le t < 178.9\mathrm{M}, \\ 0.00105, & t \ge 178.9\mathrm{M}. \end{cases} \tag{39}$$

For FFHQ-64 and AFHQv2-64, we use the same initial EDM2 schedule but adopt a more aggressive late-stage reduction:

$$\eta_{\mathrm{FFHQ/AFHQ}}(t) = \begin{cases} 0.0070, & 35.7\mathrm{M} \le t < 37.7\mathrm{M}, \\ 0.0040, & 37.7\mathrm{M} \le t < 42.9\mathrm{M}, \\ 0.0025, & 42.9\mathrm{M} \le t < 50.1\mathrm{M}, \\ 0.0018, & 50.1\mathrm{M} \le t < 51.0\mathrm{M}, \\ 0.0010, & 51.0\mathrm{M} \le t < 53.0\mathrm{M}, \\ 0.0006, & 53.0\mathrm{M} \le t < 55.0\mathrm{M}, \\ 0.0005, & 55.0\mathrm{M} \le t < 57.0\mathrm{M}, \\ 0.0003, & 57.0\mathrm{M} \le t < 59.0\mathrm{M}, \\ 0.00018, & 59.0\mathrm{M} \le t < 62.0\mathrm{M}, \\ 0.00012, & 62.0\mathrm{M} \le t < 92.0\mathrm{M}, \\ 0.00010, & 92.0\mathrm{M} \le t < 132.0\mathrm{M}, \\ 0.00009, & 132.0\mathrm{M} \le t < 162.0\mathrm{M}, \\ 0.00008, & t \ge 162.0\mathrm{M}. \end{cases} \tag{40}$$

The purpose of the closely spaced learning-rate adjustments in the later stages is to stabilize the final optimization trajectory rather than introduce additional independent contributions. We observe that once the learning rate is reduced to a sufficiently small range, further decreasing it has limited impact on the final FID. Therefore, the key factor is not the exact final learning-rate value, but rather introducing a sufficiently early and sufficiently strong transition from the high-learning-rate exploration stage to the low-learning-rate refinement stage.




| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Number of GPUs | 4 | Learning rate max($\alpha_{ref}$) | 0.017 | Adam $\beta_1$ | 0.9 | FID | 1.70 |
| Minibatch size | 512 | Learning rate decay($t_{ref}$) | 20000 | Adam $\beta_2$ | 0.99 | EMA length($\sigma_{rel}$) | 0.090 |
| Duration(Mimg) | 60 | Learning rate rampup(Mimg) | 1.0 | Loss scaling | 1 | Model capacity(Mparams) | 5.4 |
| Mixed-precision(FP16) | full | Noise distribution mean($P_{mean}$) | -1.2 | Attention res | 16 | Channel multiplier | 128 |
| Dropoutprobability | 0.13 | Noise distribution std. ($P_{std}$) | 1.2 | Attention blocks | 6 | Channels per resolution | 2-2-2 |

Figure 7: Network architecture and configuration of Configuration D on CIFAR-10.

Compared with cosine annealing, piecewise annealing provides more direct control over the timing and magnitude of learning-rate reductions. Therefore, we adopt piecewise annealing in all subsequent experiments, with the learning rates fixed according to the schedules described above. The detailed configurations of Configuration C for CIFAR-10, FFHQ-64, and AFHQv2-64 are provided in Table 6; all unspecified parameters remain the same as those in Configuration B. For Configuration F on CIFAR-10, the stability-oriented modifications reduce the convergence speed compared with the preceding configuration. Without further adjusting the learning-rate schedule, these modifications already reduce late-stage FID degradation and fluctuations, with a best FID of approximately 1.67. We then find that moderately increasing the learning rate during the intermediate training stages can slightly improve the convergence speed and FID while preserving the extended high-performance regime. Specifically, we modify the learning rate as follows:

$$\eta_{\mathrm{F}}(t) = \begin{cases} 0.0027, & 50.0\mathrm{M} \le t < 53.9\mathrm{M}, \\ 0.0023, & 53.9\mathrm{M} \le t < 63.9\mathrm{M}, \\ 0.0020, & 63.9\mathrm{M} \le t < 78.9\mathrm{M}, \\ 0.0018, & 78.9\mathrm{M} \le t < 88.9\mathrm{M}, \\ 0.0015, & 88.9\mathrm{M} \le t < 108.9\mathrm{M}. \end{cases} \tag{41}$$

These are the only learning-rate stages modified for Configuration F on CIFAR-10; all other stages and training settings remain unchanged.

### B.4 Weight Centralization and Its Gradient Dynamics (Configuration D)

The weight centralization introduced in Configuration D is applied on top of the weight-normalized convolution layers of EDM2. We therefore analyze the combined effect of centering and normalization on the resulting parameter gradients.

Let $\mathbf{w}_i \in \mathbb{R}^{1\times n}$ denote the raw weight row vector, and let $\mathbf{1} \in \mathbb{R}^{n\times 1}$ denote the $n$-dimensional all-ones vector. We first define the centered weight vector as

$$\tilde{\mathbf{w}}_i = \mathbf{w}_i - \mathrm{mean}(\mathbf{w}_i)\mathbf{1}^T = \mathbf{w}_i - \frac{1}{n}(\mathbf{w}_i\mathbf{1})\mathbf{1}^T. \tag{42}$$

This can equivalently be written as a linear projection,

$$\tilde{\mathbf{w}}_i = \mathbf{w}_i\mathbf{P}_c, \qquad \mathbf{P}_c = \mathbf{I}_n - \frac{1}{n}\mathbf{1}\mathbf{1}^T, \tag{43}$$

where $\mathbf{P}_c$ is the symmetric projection matrix onto the subspace orthogonal to the all-ones vector. The normalized weight vector is then given by

$$\hat{\mathbf{w}}_i = \frac{\tilde{\mathbf{w}}_i}{\|\tilde{\mathbf{w}}_i\|_2} = \frac{\mathbf{w}_i - \mathrm{mean}(\mathbf{w}_i)\mathbf{1}^T}{\|\mathbf{w}_i - \mathrm{mean}(\mathbf{w}_i)\mathbf{1}^T\|_2}. \tag{44}$$

By the multivariate chain rule, the gradient of the loss $L$ with respect to $\mathbf{w}_i$ is expressed as

$$\nabla_{\mathbf{w}_i} L = \left(\nabla_{\tilde{\mathbf{w}}_i}\hat{\mathbf{w}}_i \cdot \nabla_{\mathbf{w}_i}\tilde{\mathbf{w}}_i\right)^T \nabla_{\hat{\mathbf{w}}_i} L. \tag{45}$$

Since $\tilde{\mathbf{w}}_i = \mathbf{w}_i\mathbf{P}_c$ is a linear transformation,

$$\nabla_{\mathbf{w}_i}\tilde{\mathbf{w}}_i = \mathbf{P}_c. \tag{46}$$

Representing

$$\hat{\mathbf{w}}_i = \frac{\mathbf{f}(\tilde{\mathbf{w}}_i)}{g(\tilde{\mathbf{w}}_i)}, \tag{47}$$

with

$$\mathbf{f}(\tilde{\mathbf{w}}_i) = \tilde{\mathbf{w}}_i, \qquad g(\tilde{\mathbf{w}}_i) = \|\tilde{\mathbf{w}}_i\|_2, \tag{48}$$

we apply the vector quotient rule:

$$\nabla_{\tilde{\mathbf{w}}_i}\left(\frac{\mathbf{f}}{g}\right) = \frac{g\cdot\nabla_{\tilde{\mathbf{w}}_i}\mathbf{f} - \mathbf{f}^T\cdot\nabla_{\tilde{\mathbf{w}}_i}g}{g^2}. \tag{49}$$

Here,

$$\nabla_{\tilde{\mathbf{w}}_i}\mathbf{f} = \mathbf{I}_n, \qquad \nabla_{\tilde{\mathbf{w}}_i}g = \frac{\tilde{\mathbf{w}}_i}{\|\tilde{\mathbf{w}}_i\|_2} = \hat{\mathbf{w}}_i. \tag{50}$$

Substituting these components into the chain rule yields

$$\nabla_{\mathbf{w}_i} L = \left(\frac{\|\tilde{\mathbf{w}}_i\|_2\mathbf{I}_n - \tilde{\mathbf{w}}_i^T\left(\frac{\tilde{\mathbf{w}}_i}{\|\tilde{\mathbf{w}}_i\|_2}\right)}{\|\tilde{\mathbf{w}}_i\|_2^2}\mathbf{P}_c\right)^T \nabla_{\hat{\mathbf{w}}_i} L \tag{51}$$

$$= \left(\frac{1}{\|\tilde{\mathbf{w}}_i\|_2}\left(\mathbf{I}_n - \frac{\tilde{\mathbf{w}}_i^T\tilde{\mathbf{w}}_i}{\|\tilde{\mathbf{w}}_i\|_2^2}\right)\mathbf{P}_c\right)^T \nabla_{\hat{\mathbf{w}}_i} L \tag{52}$$

$$= \left(\frac{1}{\|\tilde{\mathbf{w}}_i\|_2}\left(\mathbf{I}_n - \hat{\mathbf{w}}_i^T\hat{\mathbf{w}}_i\right)\left(\mathbf{I}_n - \frac{1}{n}\mathbf{1}\mathbf{1}^T\right)\right)^T \nabla_{\hat{\mathbf{w}}_i} L \tag{53}$$

$$= \frac{1}{\|\tilde{\mathbf{w}}_i\|_2}\left(\mathbf{I}_n - \hat{\mathbf{w}}_i^T\hat{\mathbf{w}}_i\right)\left(\mathbf{I}_n - \frac{1}{n}\mathbf{1}\mathbf{1}^T\right)\nabla_{\hat{\mathbf{w}}_i} L. \tag{54}$$

Defining the orthogonal projection matrix onto the subspace orthogonal to the centered normalized weight vector as

$$\mathbf{P}_{\hat{\mathbf{w}}_i^\perp} = \mathbf{I}_n - \hat{\mathbf{w}}_i^T\hat{\mathbf{w}}_i, \tag{55}$$

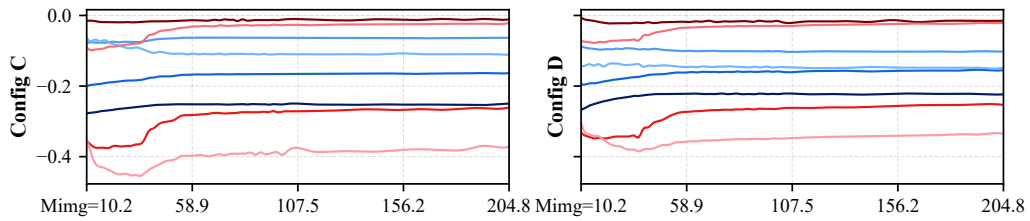
the gradient can be written compactly as

$$\boxed{\nabla_{\mathbf{w}_i} L = \frac{1}{\|\tilde{\mathbf{w}}_i\|_2}\mathbf{P}_{\hat{\mathbf{w}}_i^\perp}\mathbf{P}_c\nabla_{\hat{\mathbf{w}}_i} L}. \tag{56}$$

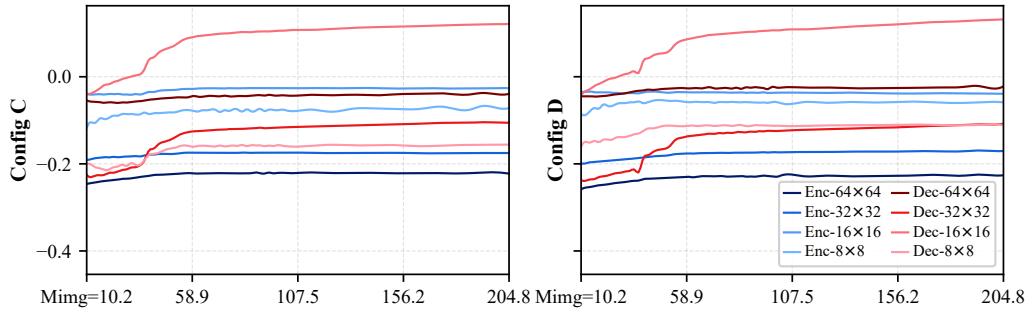
This expression reveals two projection effects introduced by centered weight normalization. First, the projection $\mathbf{P}_c$ removes the gradient component along the all-ones direction. Second, the projection $\mathbf{P}_{\hat{\mathbf{w}}_i^{\perp}}$ removes the gradient component along the direction of the centered weight vector $\tilde{\mathbf{w}}_i$. Consequently, the resulting gradient satisfies

$$\boxed{\langle \mathbf{1}, \nabla_{\mathbf{w}_i} L \rangle = 0, \qquad \langle \tilde{\mathbf{w}}_i, \nabla_{\mathbf{w}_i} L \rangle = 0.} \tag{57}$$

The first condition prevents the gradient update from introducing a nonzero mean component into the weight vector, while the second indicates that the gradient is orthogonal to the current centered weight direction. The gradient magnitude is additionally scaled by $1/\|\tilde{\mathbf{w}}_i\|_2$, which differs from the corresponding scaling factor in standard weight normalization.




(a) AFHQV2-64




(b) FFHQ-64

Figure 8: Activation mean trajectories at different network depths from 10M to 200M training images for Configurations C and D on FFHQ-64 and AFHQv2-64. Compared with the corresponding results without activation centralization, the centralized models exhibit only a mild tendency for activation means to move toward zero.

Table 6: Training configurations used for the Configuration C on different datasets.

| **Configuration** | **CIFAR-10** | **AFHQv2-64** | **FFHQ-64** |
|---|---|---|---|
| Learning rate max | 0.017 | 0.017 | 0.017 |
| Learning rate decay | 20000 | 20000 | 20000 |
| Minibatch size | 512 | 256 | 256 |
| LR ramp-up (Mimg) | 1 | 1.5 | 1.5 |
| Dropout probability | 13% | 25% | 5% |
| Channel multiplier | 128 | 128 | 128 |
| Channels per resolution | 2-2-2 | 1-2-2-2 | 1-2-2-2 |
| Duration(Mimg) | 60 | 60 | 90/140 |
| EMA length | 0.090 | 0.010 | 0.120/0.170 |
| Augment probability | 12% | 15% | 15% |

**Modified forced weight normalization.** The gradient expression above also motivates a corresponding modification to the forced weight normalization used in EDM2. In the original formu-

lation, forced weight normalization constrains the weight norm before the forward pass, thereby keeping the effective learning rate independent of weight magnitude.

With weight centralization, however, the centered weight norm becomes the quantity involved in both the forward normalization and the gradient scaling. We therefore apply forced weight normalization to the centered weight norm:

$$\boxed{\mathbf{w}_i \leftarrow \frac{\sqrt{d}}{\|\tilde{\mathbf{w}}_i\|_2}\tilde{\mathbf{w}}_i, \qquad \tilde{\mathbf{w}}_i = \mathbf{w}_i - \mathrm{mean}(\mathbf{w}_i)\mathbf{1}^T.} \tag{58}$$

This modification keeps the forced normalization constraint consistent with the norm that explicitly determines the gradient scaling under centered weight normalization. All training settings in Configuration D remain unchanged from Configuration C, except that the EMA length for the best FID on AFHQv2-64 is set to $0.035$, while for FFHQ-64, the best FID is obtained at $200\mathrm{M}$ training images with an EMA length of $0.190$. architecture is illustrated in Figure 7.

### B.5 Adaptive Frequency-aware Reconstruction Objective (Configuration E)

#### B.5.1 Function-Space Equivalence under Invertible Linear Transformation

We analyze the relationship between the standard EDM reconstruction objective and the corresponding frequency-transformed objective in function space. In particular, we show that an invertible linear transformation preserves the Bayes-optimal denoising function while modifying the geometry of the reconstruction error.

Let $\sigma \sim p_{\mathrm{train}}$ denote the noise level, $\boldsymbol{y} \sim p_{\mathrm{data}}$ the clean sample, and $\boldsymbol{n} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ the additive Gaussian noise. The noisy observation is

$$\boldsymbol{x} = \boldsymbol{y} + \boldsymbol{n}, \tag{59}$$

and the denoising network is denoted by

$$D_\theta(\boldsymbol{x}; c_{\mathrm{noise}}(\sigma)). \tag{60}$$

Let $\lambda(\sigma) > 0$ denote the noise-dependent weighting function used by EDM, and let $\Phi$ denote an invertible linear transformation. The standard and transformed reconstruction objectives are

$$L(\theta) = \mathbb{E}_{\sigma,\boldsymbol{x},\boldsymbol{y}}\left[\lambda(\sigma)\left\|D_\theta(\boldsymbol{x}; c_{\mathrm{noise}}(\sigma)) - \boldsymbol{y}\right\|_2^2\right], \tag{61}$$

$$L_\Phi(\theta) = \mathbb{E}_{\sigma,\boldsymbol{x},\boldsymbol{y}}\left[\lambda(\sigma)\left\|\Phi\left(D_\theta(\boldsymbol{x}; c_{\mathrm{noise}}(\sigma)) - \boldsymbol{y}\right)\right\|_2^2\right]. \tag{62}$$

For a fixed $(\boldsymbol{x}, \sigma)$, the corresponding conditional risks are

$$\mathcal{R}(D_\theta \mid \boldsymbol{x}, \sigma) = \lambda(\sigma)\mathbb{E}_{\boldsymbol{y}|\boldsymbol{x},\sigma}\left[\|D_\theta - \boldsymbol{y}\|_2^2\right], \tag{63}$$

$$\mathcal{R}_\Phi(D_\theta \mid \boldsymbol{x}, \sigma) = \lambda(\sigma)\mathbb{E}_{\boldsymbol{y}|\boldsymbol{x},\sigma}\left[\|\Phi(D_\theta - \boldsymbol{y})\|_2^2\right]. \tag{64}$$

For the standard objective, differentiating with respect to the network output $D_\theta$ gives

$$\nabla_{D_\theta}\mathcal{R} = 2\lambda(\sigma)\left(D_\theta - \mathbb{E}[\boldsymbol{y} \mid \boldsymbol{x}, \sigma]\right). \tag{65}$$

Hence, the unique function-space minimizer under the standard $L_2$ reconstruction loss is the posterior mean

$$D_\theta^*(\boldsymbol{x}, \sigma) = \mathbb{E}[\boldsymbol{y} \mid \boldsymbol{x}, \sigma]. \tag{66}$$

For the transformed objective, we similarly obtain

$$\begin{aligned}\nabla_{D_\theta}\mathcal{R}_\Phi &= 2\lambda(\sigma)\Phi^\top\Phi\mathbb{E}_{\boldsymbol{y}|\boldsymbol{x},\sigma}[D_\theta-\boldsymbol{y}]\\ &= 2\lambda(\sigma)\Phi^\top\Phi\left(D_\theta-\mathbb{E}[\boldsymbol{y}\mid\boldsymbol{x},\sigma]\right).\end{aligned}\tag{67}$$

If $\Phi$ is invertible, then $\Phi^\top\Phi$ is symmetric positive definite. Therefore,

$$\nabla_{D_\theta}\mathcal{R}_\Phi=\mathbf{0}\quad\Longleftrightarrow\quad D_\theta=\mathbb{E}[\boldsymbol{y}\mid\boldsymbol{x},\sigma].\tag{68}$$

Thus, the transformed objective has exactly the same function-space optimum as the standard reconstruction objective. The transformation therefore does not alter the Bayes-optimal denoising function. Instead, it changes the geometry of the reconstruction error. In particular,

$$\|\Phi(D_\theta-\boldsymbol{y})\|_2^2=(D_\theta-\boldsymbol{y})^\top\Phi^\top\Phi(D_\theta-\boldsymbol{y}),\tag{69}$$

where $\Phi^\top\Phi$ defines a positive-definite error metric. The standard MSE corresponds to $\Phi^\top\Phi=\mathbf{I}$, whereas the transformed objective assigns different weights to different directions in the output space.

Consequently, although the two objectives share the same function-space optimum, they generally induce different gradients away from the optimum. When optimization is performed with a finite-capacity neural network and a practical optimizer such as Adam, these different gradient geometries can lead to different optimization trajectories and convergence behavior.

In our formulation, the transformation is given by

$$\Phi=\mathbf{I}+\lambda\lambda_t\lambda_\sigma\,\mathrm{BPF},\tag{70}$$

where $\Phi$ may depend on the diffusion noise level and training progress, but is independent of the network parameters. The above function-space equivalence therefore applies pointwise for each $(\sigma,t)$ whenever $\Phi$ remains invertible. We implement BPF using a Difference-of-Gaussians (DoG) filter and verify the invertibility of the resulting transformation in Appendix B.5.2.

### B.5.2 Implementation of the Band-Pass Filter

Rather than using an ideal frequency-domain band-pass filter, we implement the band-pass operator using a Difference-of-Gaussians (DoG) filter. This implementation is computationally efficient, fully differentiable, and can be directly integrated into the training pipeline without explicitly transforming the images into the frequency domain.

Specifically, the band-pass operator is defined as

$$\mathrm{BPF}(\boldsymbol{x})=G_{k_1}(\boldsymbol{x})-G_{k_2}(\boldsymbol{x}),\qquad k_1<k_2,\tag{71}$$

where $G_k(\cdot)$ denotes Gaussian smoothing with standard deviation $k$. The corresponding frequency response can be written as

$$H(\boldsymbol{\omega})=\exp\left(-\frac{k_1^2\|\boldsymbol{\omega}\|_2^2}{2}\right)-\exp\left(-\frac{k_2^2\|\boldsymbol{\omega}\|_2^2}{2}\right).\tag{72}$$

Since the two Gaussian filters have different low-pass responses, their difference suppresses both the low-frequency components shared by the two filters and the very high-frequency components attenuated by both Gaussian kernels, thereby producing a band-pass-like response.

The parameters $k_1$ and $k_2$ jointly determine the location and effective bandwidth of the emphasized frequency range. In general, decreasing the Gaussian scales shifts the emphasized frequency range toward higher frequencies, while increasing the separation between $k_1$ and $k_2$ changes the effective width and shape of the passband. Rather than fixing these parameters a priori, we treat them as

Table 7: Selected hyperparameters of the frequency-aware reconstruction objective.

| Hyperparameter | CIFAR-10 | AFHQv2-64 | FFHQ-64 |
|---|---|---|---|
| $\lambda$ | 0.1 | 0.01 | 0.005 |
| $(k_1, k_2)$ | (0.8,1.6) | (1.2,2.0) | (0.8,1.6) |
| $T_{\max}$ | $7 \times 10^7$ | $7 \times 10^7$ | $7 \times 10^7$ |

hyperparameters and perform a parameter search for each dataset. The corresponding search ranges and selected values are reported in Appendix B.5.3.

The DoG formulation provides several practical advantages over an ideal frequency-domain filter. It requires only two Gaussian filtering operations, avoids explicit Fourier-domain transformations, and remains fully differentiable with respect to the input. Consequently, the same operator can be applied directly to both the denoised prediction and the target image during training:

$$\phi(\boldsymbol{x}) = \boldsymbol{x} + \lambda\lambda_t\lambda_\sigma \left[G_{k_1}(\boldsymbol{x}) - G_{k_2}(\boldsymbol{x})\right]. \tag{73}$$

Since $H(\boldsymbol{\omega}) \geq 0$, the transformation $\Phi$ defined in Eq. equation 70 is guaranteed to be invertible.

### B.5.3 Hyperparameter Settings for Frequency-aware Objective

The proposed frequency-aware objective introduces several additional hyperparameters, including the overall frequency strength $\lambda$, the band-pass frequency range $(k_1, k_2)$, the training-progress schedule parameter $T_{\max}$, and the noise-dependent weighting parameters $P_{\text{mean}}$ and $P_{\text{std}}$.

For simplicity, $T_{\max}$, $P_{\text{mean}}$, and $P_{\text{std}}$ are fixed across all experiments. Specifically, $T_{\max}$ is set to $7 \times 10^7$, while $P_{\text{mean}}$ and $P_{\text{std}}$ follow the same values used in the EDM noise-dependent weighting scheme.

We search the overall frequency strength $\lambda$ from $\{0.005, 0.01, 0.1, 1, 10\}$ and evaluate three candidate frequency ranges: $(0.4, 1.2)$, $(0.8, 1.6)$, and $(1.2, 2.0)$. For CIFAR-10, the final FID varies by only approximately 0.01 across the searched $\lambda$ values, indicating that the frequency-aware weighting has a relatively limited effect within the explored range. The selected hyperparameter values for different datasets are summarized in Table 7.

The frequency range $(k_1, k_2)$ determines which components are emphasized by the band-pass transformation. In this work, we focus on middle-frequency components and select the corresponding ranges for each dataset. For other generation tasks with different frequency characteristics, the same framework can be extended by adjusting the frequency range and searching for an appropriate frequency strength $\lambda$. The detailed configurations of Configuration E for CIFAR-10, FFHQ-64, and AFHQv2-64 are provided in Table 8; all unspecified parameters remain the same as those in Configuration D.

Table 8: Training configurations used for the Configuration E on different datasets.

| Configuration | CIFAR-10 | AFHQv2-64 | FFHQ-64 |
|---|---|---|---|
| Learning rate max | 0.017 | 0.017 | 0.017 |
| Learning rate decay | 20000 | 20000 | 20000 |
| Minibatch size | 512 | 256 | 256 |
| LR ramp-up (Mimg) | 1 | 1.5 | 1.5 |
| Dropout probability | 13% | 25% | 5% |
| Channel multiplier | 128 | 128 | 128 |
| Channels per resolution | 2-2-2 | 1-2-2-2 | 1-2-2-2 |
| Duration(Mimg) | 55 | 60 | 100 |
| EMA length | 0.080 | 0.010 | 0.015 |
| Augment probability | 12% | 15% | 15% |

### B.6 MP Component Modifications

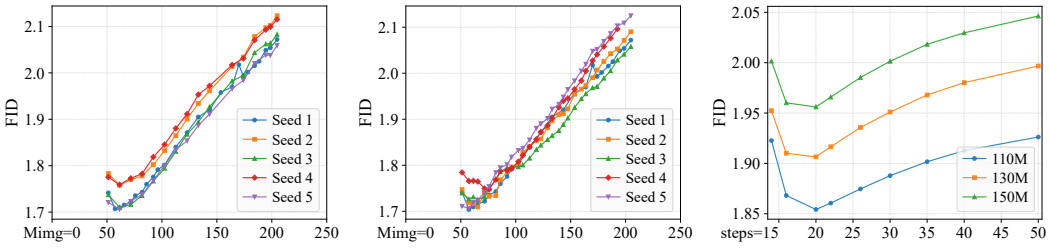
We further investigate the effects of the magnitude-preserving (MP) components used in EDM2. The MP-SiLU and MP-Fourier components are kept unchanged, while MP-Sum is replaced with the unnormalized linear interpolation

$$\mathrm{lerp}(a, b; t) = (1 - t)a + tb, \tag{74}$$

and MP-Cat is replaced with ordinary channel-wise concatenation. The resulting FID trajectory is shown in Figure 3b. These modifications delay the point at which the best FID is reached while substantially extending the subsequent high-performance regime. It is worth noting that, for the CIFAR-10 architecture, the two feature maps combined by MP-Cat have the same number of channels. Therefore, at the interpolation coefficient used in our experiments, replacing MP-Cat with ordinary concatenation is mathematically equivalent to the original MP-Cat operation. On AFHQv2-64, where the corresponding channel dimensions differ, removing MP-Cat has only a minor effect on the resulting performance.

We also examine more direct replacements of the MP components. Replacing MP-SiLU and MP-Fourier with the standard SiLU activation and conventional Fourier encoding, respectively, leads to degraded generation quality. Moreover, directly replacing MP-Sum with ordinary addition, $a + b$, causes training to fail. These observations suggest that the individual MP components cannot be simply removed without affecting the training dynamics, although the precise mechanism underlying their influence on the duration of the high-performance regime remains unclear.

### B.7 Robustness Checks




(a) (b) (c)

Figure 9: Robustness checks for the late-stage FID degradation of Configuration E. (a) FID trajectories under five sampling seeds using the same trained checkpoints. (b) FID trajectories of Configuration E trained with five training seeds. (c) FID of Config. E checkpoints at 110M, 130M, and 150M training images under a sampling-step sweep from 14 to 50 steps.

First, all sampling experiments in the main study use the same random seed to enable controlled comparisons between configurations. To determine whether the observed temporal behavior is specific to this seed, we reevaluate the FID trajectory of Config. E using four additional random seeds. As shown in Figure 9a, although the absolute FID values vary across seeds, the qualitative temporal trend remains consistent.

Second, to examine whether the observed degradation is specific to the training seed used in the main experiments, we additionally retrain Configuration E on CIFAR-10 using four different training random seeds. The resulting FID trajectories are shown in Figure 9b. Although the absolute FID values and the precise locations of the best checkpoints vary across runs, the late-stage FID degradation remains consistently observable, suggesting that this behavior is not specific to a particular training seed.

Third, because all sampling experiments use a fixed number of sampling steps, the observed late-stage degradation could in principle result from increased discretization error. The underlying probability flow ODE may contain regions of relatively large local curvature, and as training progresses, the learned denoising model may increasingly approximate this ODE. Consequently, a small fixed number of sampling steps could become insufficient to accurately resolve the learned trajectory,

leading to larger numerical errors and degraded FID. To examine this possibility, we extract Config. E checkpoints at 110M, 130M, and 150M training images and evaluate each checkpoint with a sampling-step sweep from 14 to 50 steps. As shown in Figure 9c, increasing the number of sampling steps does not recover the best FID previously achieved by Config. E, suggesting that the observed late-stage degradation cannot be explained solely by the use of too few sampling steps.

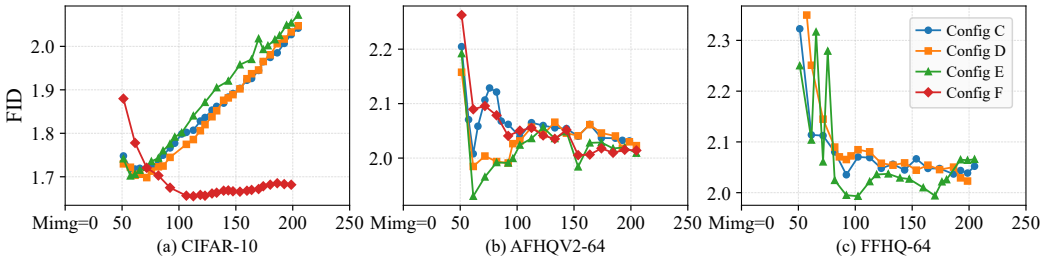



Figure 10: FID trajectories of Configurations C, D, E, and F on CIFAR-10, AFHQv2-64, and FFHQ-64. Configurations C, D, and E exhibit similar overall trends, with FID degradation or oscillation emerging after reaching their respective best performance. Configuration F shows improved persistence of the high-performance regime. Lower FID indicates better performance.

# C Loss Landscape Analysis

## C.1 Experimental Settings and Parameter Subspace Selection

To analyze the local optimization geometry induced by the EDM2 architecture, we compute the leading eigenvalues of the Hessian at different stages of training for both EDM2 and the EDM-style control model. The two models are evaluated using the same fixed probe dataset, the same noise levels, and the same parameter subspace, so that changes in the measured Hessian spectrum can be attributed primarily to the evolution of the model parameters rather than to changes in the evaluation conditions.

**Model configurations.** The EDM2 model used in this analysis is based on Configuration C with the uncertainty-based multi-task objective removed. The EDM-style control model is then constructed by removing the EDM2-specific magnitude preserving (MP) operators, weight normalization, forced weight normalization, and pixel normalization, and restoring group normalization. This construction aims to isolate the effect of the architectural normalization and parameterization changes introduced by EDM2 while keeping the remaining training configuration as consistent as possible. After removing the aforementioned components, directly retaining the EDM2 initialization can produce excessively large activation magnitudes during the initial stage of training and cause the training to fail. We therefore use the initialization scheme of EDM for the control model. The control model is also trained with a fixed learning rate(while retaining the zero initialization of the output gain; the resulting EDM-style control model achieves a FID of $1.77 \pm 0.02$ across five training seeds). To avoid having the difference in initialization dominate the curvature comparison, we only perform the Hessian analysis after 30M training images.

**EMA snapshots.** Rather than evaluating the instantaneous training parameters, we perform all Hessian analyses on short-length EMA snapshots. Specifically, we reconstruct post-hoc power-law EMA snapshots with an EMA length of 0.010. This short EMA length filters out high-frequency parameter fluctuations while introducing only limited temporal lag, making the resulting snapshots a more stable and representative target for measuring the local loss landscape than instantaneous training parameters.

**Parameter subspace.** Although the complete networks contain convolutional layers, attention parameters, embedding layers, and several functional learnable scaling parameters, we restrict the Hessian analysis to the `conv_res0` and `conv_res1` convolutional layers in the residual paths.

This parameter subspace contains approximately $47.8$M trainable parameters, compared with approximately $54$M parameters in the complete network, and therefore accounts for the majority of the model parameters.

This restriction is motivated by both functional correspondence and parameterization consistency. The selected convolutional layers perform the main feature extraction and feature transformation operations along the residual paths and are structurally shared by both model variants. In contrast, the remaining parameters play substantially different functional roles. Some convolutional layers primarily adjust feature dimensions or channel numbers, while the attention module contains $W_{qkv}$ and output projection parameters whose effect on the network output follows a different computational structure involving attention scores. Similarly, the embedding layers primarily encode conditioning information, and learnable scaling factors mainly modulate the magnitude of existing features. Including these heterogeneous parameter types would mix parameter subspaces with different functional roles and potentially introduce additional differences in parameterization and scale. Importantly, the same parameter subspace is used for both EDM2 and the EDM-style control model. Thus, the restriction is not intended to select a parameter subset with favorable curvature, but rather to establish a structurally matched parameter space in which the two architectures can be compared directly. Since the selected layers already account for the vast majority of trainable parameters, their local Hessian spectrum provides a representative characterization of the dominant feature extraction path.

**Fixed probe dataset.** We evaluate the Hessian on a fixed probe CIFAR-10 dataset containing $4096$ training samples. For each sample, the clean image, class label, and noise level $\sigma$ are fixed throughout the entire analysis. The noisy input is therefore also fixed for all EMA snapshots. This ensures that Hessian measurements at different training stages are performed under identical input conditions. Due to GPU memory constraints, the $4096$ samples are processed in smaller micro-batches and accumulated across batches, analogous to gradient accumulation. This procedure computes the Hessian of the empirical loss over the full probe dataset without changing the underlying objective.

**Hessian estimation.** For each EMA snapshot, we compute Hessian-vector products in the selected parameter subspace and estimate the leading Hessian eigenvalues using the Lanczos algorithm (Golub & O'Leary, 1989; Ghorbani et al., 2019). We use $60$ Lanczos iterations and retain the top-5 eigenvalues. In particular, the largest eigenvalue $\lambda_{\max}$ is used in the scale-corrected sensitivity metric defined in Eq. 6. The random initial Lanczos vector is fixed during the main analysis, and experiments with different random seeds produce nearly identical leading eigenvalues, indicating that the observed trends are not caused by the random initialization of the Lanczos procedure. However, because the two models have different parameter scales, directly comparing $\lambda_{\max}(H)$ is insufficient. We therefore combine the Hessian curvature with the parameter scale through the relative-perturbation analysis described in Appendix C.2.

**Learning-Rate Scheduling.** One consideration is that the EDM2-style and EDM-style networks do not use identical learning-rate schedules in our comparison. This difference is closely related to the parameterization of EDM2. With forced weight normalization, the magnitude of the weights is explicitly controlled, effectively decoupling parameter magnitude from the evolution of the effective learning rate. Consequently, an explicit learning-rate schedule is required to control the optimization step size over the course of training, and the learning-rate schedule should therefore be regarded as an important component of the overall optimization behavior of EDM2. In contrast, our measurements show that the weight norms of the EDM-style control model gradually increase during training, implying a corresponding decrease in its effective learning rate even when the nominal learning rate remains fixed. Thus, the two models effectively employ different forms of learning-rate evolution: EDM2 uses explicit annealing, whereas the EDM-style model exhibits an implicit reduction in effective update scale through the growth of parameter magnitude. Nevertheless, because the optimization trajectories are not fully matched, the observed curvature evolution cannot be attributed exclusively to the parameterization or the learning-rate schedule.

A related consideration is the well-known coupling between learning rate and local curvature. For gradient descent, prior work has shown that the largest Hessian eigenvalue can approach the stability threshold $2/\eta$, a phenomenon commonly referred to as the *Edge of Stability* (Cohen et al., 2021). Although this relationship does not directly characterize Adam or imply a universal equality between

learning rate and sharpness, it motivates examining whether the different learning-rate schedules could explain the observed curvature trends. Our measurements provide evidence against this interpretation as a complete explanation. In the later training stage, the learning rates of the EDM2-style and EDM-style networks become relatively close, yet the leading Hessian eigenvalue remains substantially smaller for EDM2: for example, at approximately 200M training images, $\lambda_{\max}$ is about $0.00066$ for EDM2 compared with approximately $0.0011$ for EDM under comparable learning rates. Thus, the difference in learning rate alone cannot account for the lower curvature observed in EDM2. At the same time, the learning-rate schedule may still contribute to the temporal evolution of EDM2 curvature. In particular, we observe that the increase in $\lambda_{\max}$ becomes progressively slower as the EDM2 learning rate enters a slowly varying regime, suggesting that learning-rate decay and curvature evolution may be coupled. However, this cannot explain the decreasing $\lambda_{\max}$ observed in the EDM-style control model despite its fixed nominal learning rate. Therefore, the different learning-rate schedules may affect the evolution of local curvature, but they do not by themselves explain the persistent curvature difference between the two parameterizations. A fully controlled study with matched learning-rate trajectories would be required to disentangle these effects and is left for future work.

**Hessian Spectrum and Negative Eigenvalues.** Our analysis focuses on the maximum change in the gradient induced by a relative parameter perturbation. In the general case, this quantity is governed by the spectral norm of the Hessian, $|H| * 2 = \max_i |\lambda_i|$, rather than necessarily by the largest positive eigenvalue. In practice, away from the initialization regime, the dominant curvature directions observed in our experiments are associated with positive eigenvalues. To verify that negative curvature does not affect our choice of $\lambda * \max$, we additionally examine the smallest Hessian eigenvalue in the selected parameter subspace. Across the evaluated checkpoints, the magnitude of the most negative eigenvalue remains below 80% of the leading positive eigenvalue for both the EDM-style and EDM2-style networks. More specifically, for the EDM2-style network, only a very small number of checkpoints exhibit a magnitude of the most negative eigenvalue approaching 40% of the leading positive eigenvalue, while at nearly all other checkpoints it remains below 10%. For the EDM-style network, a very small number of checkpoints in the later training stage exhibit a magnitude approaching 77% of the leading positive eigenvalue, whereas it remains below 30% at the other checkpoints. Thus, within the parameter subspace and training range considered in this study, the spectral norm of the Hessian is effectively determined by the leading positive eigenvalue. We therefore use $\lambda_{\max}(H)$ as the primary curvature measure in our scale-corrected sensitivity analysis.

**Optimizer Considerations.** The proposed scale-corrected sensitivity is a local geometric quantity that characterizes how strongly the gradient changes under a relative perturbation of the parameters, and is therefore independent of the specific optimizer used to update the parameters. Although our models are trained with Adam, whose updates depend on exponentially weighted first- and second-order gradient moments, the local gradient sensitivity remains relevant because a smaller gradient variation under the same relative parameter perturbation indicates a more slowly varying local gradient field. This can reduce the discrepancy between the gradient evaluated at the current parameters and the gradients encountered during a finite parameter update, thereby reducing local update error associated with gradient variation. However, the actual Adam update is additionally affected by its moment estimates and adaptive normalization. Therefore, our scale-corrected sensitivity should not be interpreted as a direct measure of Adam's update magnitude or as a complete characterization of Adam's optimization dynamics. Rather, it provides an optimizer-independent measure of local gradient sensitivity that can help characterize the optimization geometry underlying the observed training behavior.

### C.2 Scale-Corrected Loss Landscape Analysis

Directly comparing the largest Hessian eigenvalue between two networks with different parameter scales can lead to an incomplete interpretation. The relevant question for optimization is not only how large the local curvature is under an absolute parameter perturbation, but how strongly the gradient changes when the parameters are perturbed by the same fraction of their respective scales.

Consider the gradient of a loss function $L(\theta)$ at a parameter point $\theta$. For a sufficiently small parameter perturbation $\Delta\theta$, the first-order Taylor expansion of the gradient gives

$$\nabla L(\theta + \Delta\theta) \approx \nabla L(\theta) + H(\theta)\Delta\theta, \tag{75}$$

where $H(\theta)$ is the Hessian matrix. Therefore, the resulting change in the gradient is approximately

$$\Delta\nabla L \triangleq \nabla L(\theta + \Delta\theta) - \nabla L(\theta) \approx H(\theta)\Delta\theta. \tag{76}$$

Taking the Euclidean norm and using the spectral norm of the Hessian, we obtain

$$\|\Delta\nabla L\|_2 \leq \|H(\theta)\|_2\|\Delta\theta\|_2 = \lambda_{\max}(H)\|\Delta\theta\|_2, \tag{77}$$

where the last equality holds for a symmetric Hessian whose largest eigenvalue is the relevant spectral radius. The upper bound is attained when $\Delta\theta$ is aligned with the eigenvector corresponding to $\lambda_{\max}(H)$.

To eliminate the effect of different absolute parameter scales, we consider a relative perturbation of magnitude $\epsilon$:

$$\frac{\|\Delta\theta\|_2}{\|\theta\|_2} = \epsilon, \qquad \|\Delta\theta\|_2 = \epsilon\|\theta\|_2. \tag{78}$$

Substituting Eq. equation 78 into Eq. equation 77 gives

$$\|\Delta\nabla L\|_2 \leq \epsilon\lambda_{\max}(H)\|\theta\|_2. \tag{79}$$

This motivates the scale-corrected local sensitivity measure

$$\mathcal{S}(H, \theta) \triangleq \lambda_{\max}(H)\|\theta\|_2 \tag{80}$$

which characterizes the maximum gradient change induced by a unit relative parameter perturbation, up to the common factor $\epsilon$. This quantity characterizes the worst-case change in the Euclidean gradient under a fixed relative parameter perturbation; it is not intended to approximate the actual parameter-update magnitude produced by Adam. Consequently, for two models $A$ and $B$, their relative loss-landscape sensitivity can be compared through

$$\rho_{A/B} = \frac{\lambda_{\max}(H_A)\|\theta_A\|_2}{\lambda_{\max}(H_B)\|\theta_B\|_2}. \tag{81}$$

In particular, we use

$$\rho = \frac{\lambda_{\max}(H_{\mathrm{EDM2}})\|\theta_{\mathrm{EDM2}}\|_2}{\lambda_{\max}(H_{\mathrm{EDM}})\|\theta_{\mathrm{EDM}}\|_2} \tag{82}$$

throughout our analysis. A value $\rho < 1$ indicates that EDM2 exhibits a smaller maximum gradient change than the EDM-style control model under the same relative parameter perturbation.

The use of a relative perturbation is important because the two models exhibit substantially different weight scales. Suppose the parameters are rescaled according to $\theta' = \alpha\theta$, while the corresponding Hessian under this parameter rescaling transforms as

$$H' = \frac{1}{\alpha^2}H. \tag{83}$$

Then

$$\lambda_{\max}(H')\|\theta'\|_2 = \frac{1}{\alpha^2}\lambda_{\max}(H) \cdot \alpha\|\theta\|_2 = \frac{1}{\alpha}\lambda_{\max}(H)\|\theta\|_2. \tag{84}$$

This shows an important distinction from a naive comparison of $\lambda_{\max}(H)$: the quantity $\lambda_{\max}(H)\|\theta\|_2$ is appropriate for comparing the gradient response under a relative perturbation, but it is not itself strictly invariant under an arbitrary reparameterization $\theta \to \alpha\theta$. Rather, its role is to convert the absolute Hessian curvature into the gradient-change scale associated with a fixed relative perturbation. This is precisely the quantity relevant to the comparison performed in this work.

**RMS representation of parameter scale.** In practice, we measure the RMS of the weights rather than explicitly reporting their full Euclidean norm. For a weight tensor containing $d$ scalar parameters,

$$\mathrm{RMS}(\theta) = \sqrt{\frac{1}{d}\sum_{i=1}^{d}\theta_i^2}, \qquad \|\theta\|_2 = \sqrt{d}\,\mathrm{RMS}(\theta). \tag{85}$$

In the selected parameter subspace of our experiments, the corresponding convolutional layers have the same input and output channel dimensions, so that the parameter dimensionality $d$ is identical across the corresponding layers of the two models. Consequently, the common factor $\sqrt{d}$ cancels when forming the relative parameter-scale ratio:

$$\frac{\|\theta_{\mathrm{EDM2}}\|_2}{\|\theta_{\mathrm{EDM}}\|_2} = \frac{\mathrm{RMS}(\theta_{\mathrm{EDM2}})}{\mathrm{RMS}(\theta_{\mathrm{EDM}})}. \tag{86}$$

Therefore, the RMS measurements are sufficient for the scale correction used in our experiments, without requiring the full parameter norms to be explicitly reported.

Finally, normalization in both EDM2 and the EDM-style baseline can affect the effective perturbation experienced by the network. In EDM2, the forced weight normalization explicitly renormalizes the weight vectors before each forward pass, so a perturbation component parallel to the current weight vector can be partially or completely removed by the subsequent renormalization. However, a related scale invariance is also present in the EDM-style baseline through GroupNorm. In particular, for a positive scalar $c$, GroupNorm satisfies $\mathrm{GN}(cWx) = \mathrm{GN}(Wx)$, so a pure positive rescaling of the weights can likewise have little or no direct effect on the normalized activation.

Therefore, the removal of radial parameter perturbations should not be regarded as an EDM2-specific source of reduced sensitivity. Rather, both normalization schemes introduce parameter-to-function invariances that can affect how parameter perturbations are manifested in the network. Our scale-corrected sensitivity is instead defined over the worst-case perturbation satisfying $|\Delta\theta|_2 = \epsilon|\theta|_2$, and therefore does not rely on the behavior of any particular perturbation direction. The observed difference in $\mathcal{S}(H, \theta)$ should consequently be interpreted as a property of the local geometry of the two parameterizations under the specified relative perturbation model, rather than being attributed to the removal of radial perturbations by forced weight normalization alone.